make: Nothing to be done for `all'

Asked 10 years, 9 months ago Modified 1 month ago Viewed 302k times

135

23

I am going through an eg pgm to create a make file.

http://mrbook.org/tutorials/make/

My folder eg_make_creation contains the following files,

```
desktop:~/eg_make_creation$ ls
factorial.c  functions.h  hello  hello.c  main.c  Makefile
```

Makefile

```
# I am a comment, and I want to say that the variable CC will be
# the compiler to use.
CC=gcc
# Hwy!, I am comment no.2. I want to say that CFLAGS will be the
#options I'll pass to the compiler
CFLAGS=-c -Wall

all:hello

hello:main.o factorial.o hello.o
  $(CC) main.o factorial.o hello.o -o hello

main.o:main.c
  $(CC) $(CFLAGS) main.c

factorial.o:factorial.c
  $(CC) $(CFLAGS) factorial.c

hello.o:hello.c
  $(CC) $(CFLAGS) hello.c

clean:
  rm -rf *o hello
```

error:

```
desktop:~/eg_make_creation$ make all
make: Nothing to be done for `all'.
```

Please help me understand to compile this program.

c makefile

Aaron McDaid
25.8k ● 9 ● 61 ● 86

Angus
11.6k ● 28 ● 87 ● 148

22 Try doing a "make clean" followed by a "make all" – Paul R Dec 19, 2011 at 12:56

14 That's not an error, it just means `hello` is up to date. Change `clean` to `rm -f *.o hello` before it does something unexpected, then run `make clean all` and see if that works. – Mat Dec 19, 2011 at 12:57

1 You should also add `.phony: all clean`, since `all` and `clean` aren't file names. – Kerrek SB Dec 19, 2011 at 12:58

3 Don't put the -c in your CFLAGS. – wildplasser Dec 19, 2011 at 12:59

10 Answers

Sorted by:
Trending sort available
Highest score (default)

167

Sometimes "Nothing to be done for all" error can be caused by spaces before command in makefile rule instead of tab. Please ensure that you use tabs instead of spaces inside of your rules.

```
all:
<\t>$(CC) $(CFLAGS) ...
```

instead of

```
all:
    $(CC) $(CFLAGS) ...
```

Please see the GNU make manual for the rule syntax description: https://www.gnu.org/software/make/manual/make.html#Rule-Syntax

Share Follow

edited Jan 14, 2019 at 7:40

answered May 25, 2013 at 13:07
VirtualVDX
2,091 ● 1 ● 12 ● 13

obj-m += hello.o all: </t>make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules clean: make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean – Narendra Jaggi Jun 21, 2015 at 8:14

is the above file is valid file?? – Narendra Jaggi Jun 21, 2015 at 8:15

In the context of parent-child makefiles, sometimes "Nothing to be done for all" error can be caused by child targets being incorrectly marked declared .PHONY in the parent Makefile. – donhector Apr 27, 2018 at 23:57

I had `all :  src/server/mod_wsgi.la`, which I changed to `all : </t> src/server/mod_wsgi.la`. I now get the error : `make: execvp: src/server/mod_wsgi.la: Permission denied Makefile:29:`

1 @eryanbnu please make sure your rules are following the rule syntax: gnu.org/software/make/manual/make.html#Rule-Syntax. – VirtualVDX Apr 24 at 13:21

37

Remove the `hello` file from your folder and try again.

The `all` target depends on the `hello` target. The `hello` target first tries to find the corresponding file in the filesystem. If it finds it and it is up to date with the dependent files—there is nothing to do.

Share Follow

edited Nov 9, 2015 at 10:24
fuz
84.8k ●24 ●190 ●337

answered Dec 19, 2011 at 12:56
weekens
7,884 ●6 ●43 ●60

obj-m += hello.o all: </t>make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules clean: make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean – Narendra Jaggi Jun 21, 2015 at 8:15

is the above one is a valid make file? – Narendra Jaggi Jun 21, 2015 at 8:16

21

When you just give make, it makes the first rule in your makefile, i.e "all". You have specified that "all" depends on "hello", which depends on main.o, factorial.o and hello.o. So 'make' tries to see if those files are present.

If they are present, 'make' sees if their dependencies, e.g. main.o has a dependency main.c, have changed. If they have changed, make rebuilds them, else skips the rule. Similarly it recursively goes on building the files that have changed and finally runs the top most command, "all" in your case to give you a executable, 'hello' in your case.

If they are not present, make blindly builds everything under the rule.

Coming to your problem, it isn't an error but 'make' is saying that every dependency in your makefile is up to date and it doesn't need to make anything!

Share Follow

answered Dec 19, 2011 at 13:28
Chethan Ravindranath
1,941 ●1 ●15 ●28

18

Make is behaving correctly. `hello` already exists and is not older than the `.c` files, and therefore there is no more work to be done. There are four scenarios in which make will need to (re)build:

- If you modify one of your `.c` files, then it will be newer than `hello`, and then it will have to rebuild when you run make.
- If you delete `hello`, then it will obviously have to rebuild it

comment about `rm -f *.o hello`

Share Follow answered Dec 19, 2011 at 13:57

community wiki
Aaron McDaid

5

I think you missed a tab in 9th line. The line following **all:hello** must be a blank tab. Make sure that you have a blank tab in 9th line. It will make the interpreter understand that you want to use default recipe for makefile.

Share Follow

answered Jan 11, 2018 at 7:29
abhinav0653
51 ● 1 ● 2

5

That is not an error; the make command in unix works based on the timestamps. I.e let's say if you have made certain changes to `factorial.cpp` and compile using `make` then make shows the information that only the `cc -o factorial.cpp` command is executed. Next time if you execute the same command i.e `make` without making any changes to any file with `.cpp` extension the compiler says that the output file is up to date. The compiler gives this information until we make certain changes to any `file.cpp`.

The advantage of the `makefile` is that it reduces the recompiling time by compiling the only files that are modified and by using the object (`.o`) files of the unmodified files directly.

Share Follow

edited Jun 29, 2020 at 5:49
tripleee
163k ● 27 ● 243 ● 295

answered Jan 15, 2012 at 12:58
muneshwar
51 ● 1

4

Using the comment from Paul R, I found that

```
make clean
```

followed by

```
make
```

or

```
make all
```

fixed my problem.

0

I arrived at this peculiar, hard-to-debug error through a different route. My trouble ended up being that I was using a pattern rule in a build step when the target and the dependency were located in distinct directories. Something like this:

```
foo/apple.o: bar/apple.c $(FOODEPS)

%.o: %.c
    $(CC) $< -o $@
```

I had several dependencies set up this way, and was trying to use one pattern recipe for them all. Clearly, a single substitution for "%" isn't going to work here. I made explicit rules for each dependency, and I found myself back among the puppies and unicorns!

```
foo/apple.o: bar/apple.c $(FOODEPS)
    $(CC) $< -o $@
```

Hope this helps someone!

Share Follow

answered Jul 13, 2019 at 23:40

sfaleron

41 ● 3

0

I was trying to install libuv on Ubuntu and i also got the error make: `Nothing to be done for 'all'`. As i see it, using make gives two ways to solve the problem, one for check and one for install. But i found a workaround still use the sudo `make check` command - it helps to read all the error messages before deciding on further actions. Basically, i've introduced a regression that makes the update workaround inefficient. This error comes from make however, the workaround from install fixes this, just try to run sudo `make install` and see what happens. The make command will be a local optimization at the expense of the overall result of `check/install` - c'est ma façon de parler. I believe i have narrowed down the problem considerably: in the first case after **check** i have "`FAIL: test/run-tests`" and in the second after **install** i get `"specify the full pathname of the library, or use the '-LLIBDIR'"` This argument to check/install can be a list object to store information about completed installations. So install reports partial success when nothing actually happened.

Try running the commands from root:

```
cd your_program
sh autogen.sh
./configure
make
make check
make install
```

```
Libraries have been installed in:
/usr/local/lib
```

Share Follow

edited Feb 10 at 4:45

answered Feb 9 at 21:59
Fithe_Xanki
97 ● 1 ● 4

0

In your case, I strongly feel the only and simple problem you had is that you only preprocessed your app. You did so by having the flag `-c` under `CFLAGS`.

Share Follow

answered Aug 1 at 6:33
Jared Atandi
54 ● 5

You are wrong. `-c` actually creates an object file. This is not just preprocessing. Running just the preprocessor is `-E`. BTW - "feeling" is not a valid cause for posting a quality answer. Other people already pointed to the real issue that the program is most probably already built and hence `make` has nothing left to do. – reichhart Aug 4 at 16:46

This does not provide an answer to the question. Once you have sufficient reputation you will be able to comment on any post; instead, provide answers that don't require clarification from the asker. - From Review – HandsomeGorilla Aug 4 at 19:00