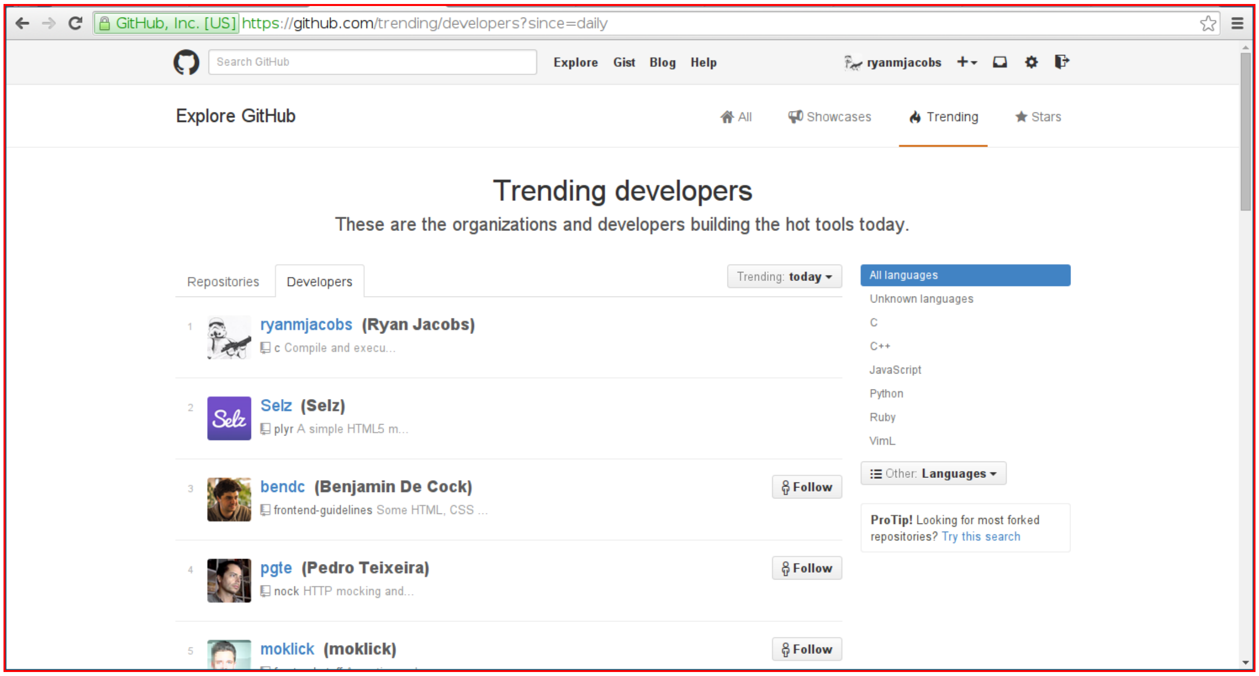
ryanmjacobs/c

In 2015, I came home from school one day and was writing some C code for fun. I can't recall what I was writing at the time, but I do remember finding myself annoyed at the length and repetitive compile-and-run cycle that I was facing. I figured that other people might have the same issue too. So I decided to write a tool that would allow people to compile-and-run C programs with greater speed and ease. At the time, the methods for running C code as scripts were convoluted and confusing. But after a few days, I came up an elegant solution to the problem and published my code online: https://github.com/ryanmjacobs/c. My tool spread like wildfire, and it has been my most successful project to date. I received over 100,000 downloads in the first 24 hours and was the number one trending developer on GitHub for 3 days. GitHub is a social coding platform that acts as the defacto hosting solution for the software world. Nearly every software developer in the world uses GitHub, so to be at the top of their site is not an easy task. Company recruiters often use GitHub to locate top developers to interview. And so even I got a few recruitment emails myself, despite being only 16 years old at the time. Most of the emails were from smaller companies, but I did get one from Nvidia — the largest consumer graphics card manufacturer in the world. I am just glad that people are still using my tool and appreciating my work. It is extremely gratifying to know that I contributed something useful to the world.

Here's me (Ryan Jacobs) trending on GitHub:



Trending developers

These are the organizations and developers building the hot tools today.

1 ryanmjacobs (Ryan Jacobs)
c Compile and execu...

2 Selz (Selz)
plyr A simple HTML5 m...

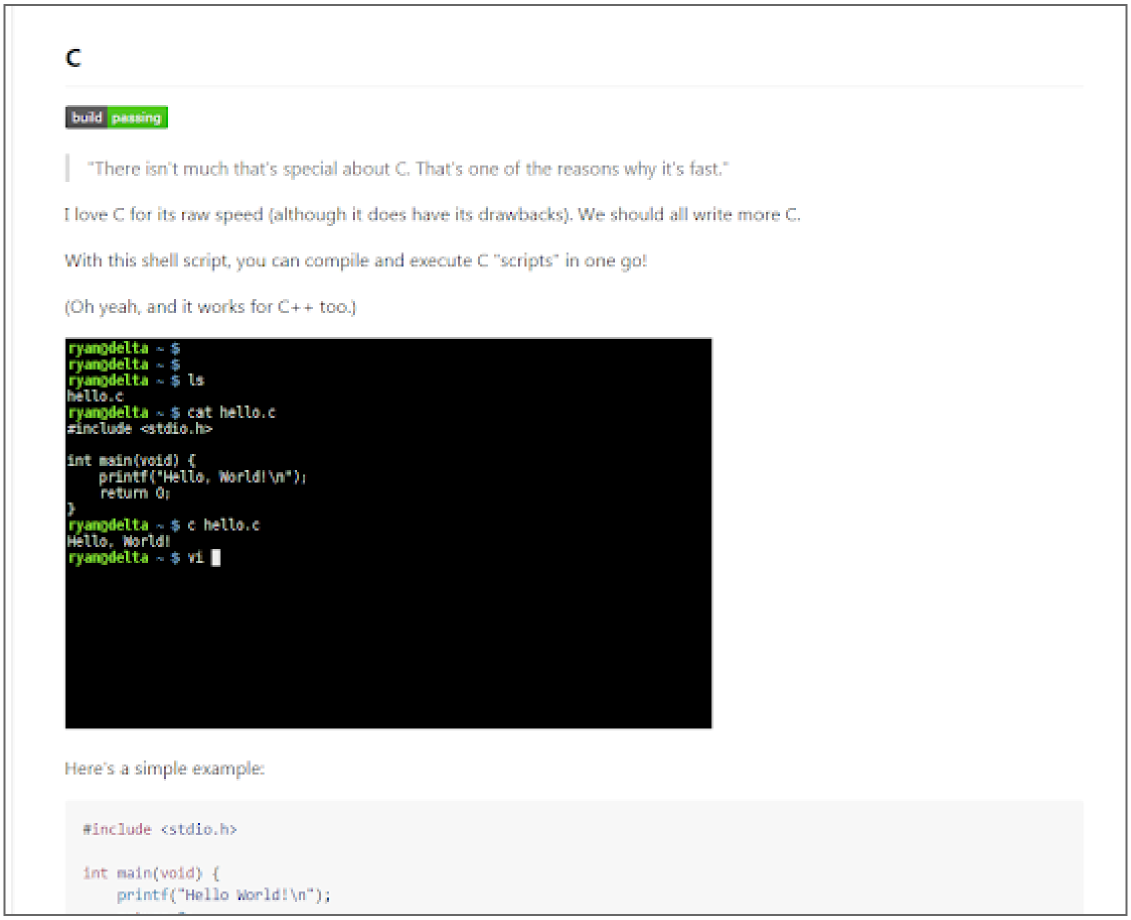
3 bendc (Benjamin De Cock) Follow
frontend-guidelines Some HTML, CSS ...

4 pgte (Pedro Teixeira) Follow
nock HTTP mocking and...

5 moklick (moklick) Follow

ProTip! Looking for most forked repositories? Try this search

My about page / usage instructions:



C

> "There isn't much that's special about C. That's one of the reasons why it's fast."

I love C for its raw speed (although it does have its drawbacks). We should all write more C.

With this shell script, you can compile and execute C "scripts" in one go!

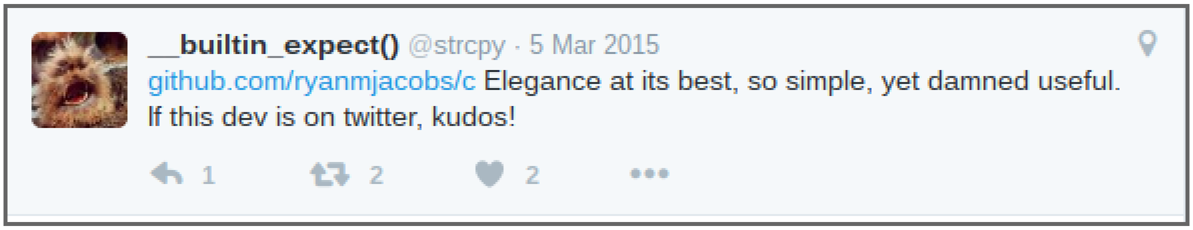
(Oh yeah, and it works for C++ too.)

```
ryan@delta ~ $
ryan@delta ~ $
ryan@delta ~ $ ls
hello.c
ryan@delta ~ $ cat hello.c
#include <stdio.h>

int main(void) {
    printf("Hello, World!\n");
    return 0;
}
ryan@delta ~ $ c hello.c
Hello, World!
ryan@delta ~ $ vi 
```

Here's a simple example:

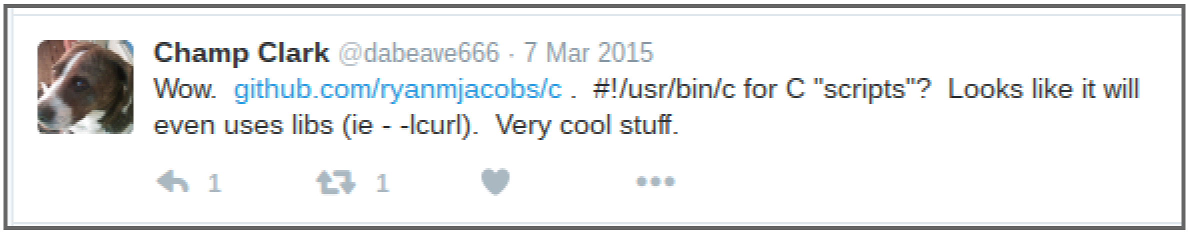
```
#include <stdio.h>

int main(void) {
    printf("Hello World!\n");
```

I received a lot of response on social networking sites, especially Twitter:



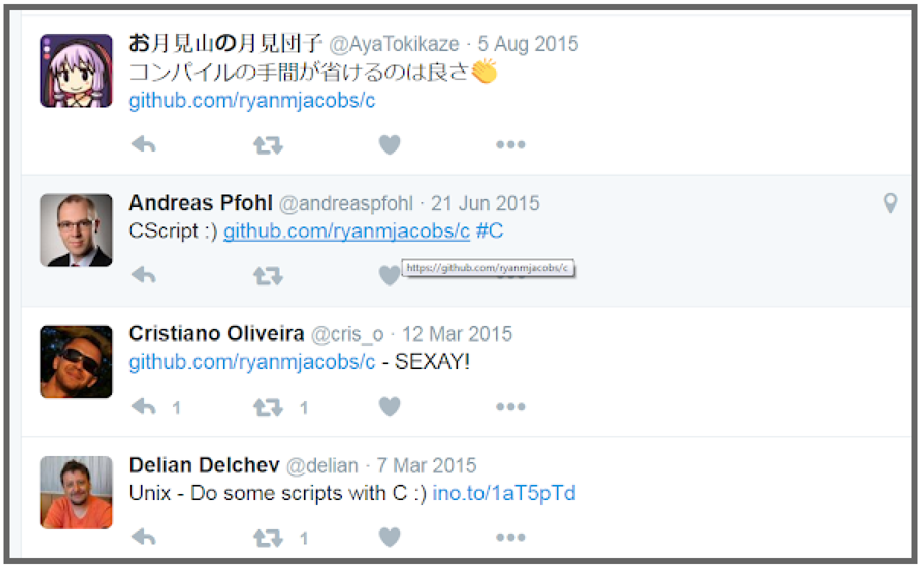
__builtin_expect() @strcpy · 5 Mar 2015

github.com/ryanmjacobs/c Elegance at its best, so simple, yet damned useful. If this dev is on twitter, kudos!



Champ Clark @dabeave666 · 7 Mar 2015

Wow. github.com/ryanmjacobs/c . #!/usr/bin/c for C "scripts"? Looks like it will even uses libs (ie - -lcurl). Very cool stuff.

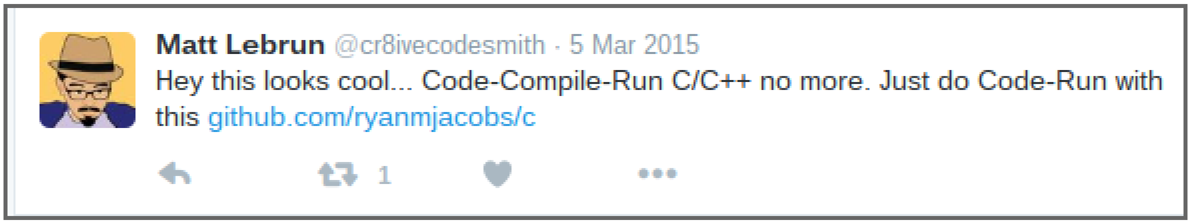


お月見山の月見団子 @AyaTokikaze · 5 Aug 2015
コンパイルの手間が省けるのは良さ👏
github.com/ryanmjacobs/c

Andreas Pfohl @andreaspfohl · 21 Jun 2015
CScript :) github.com/ryanmjacobs/c #C

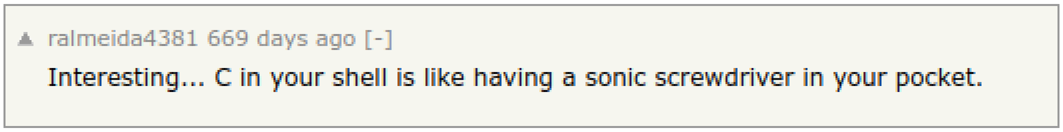
Cristiano Oliveira @cris_o · 12 Mar 2015
github.com/ryanmjacobs/c - SEXAY!

Delian Delchev @delian · 7 Mar 2015
Unix - Do some scripts with C :) ino.to/1aT5pTd



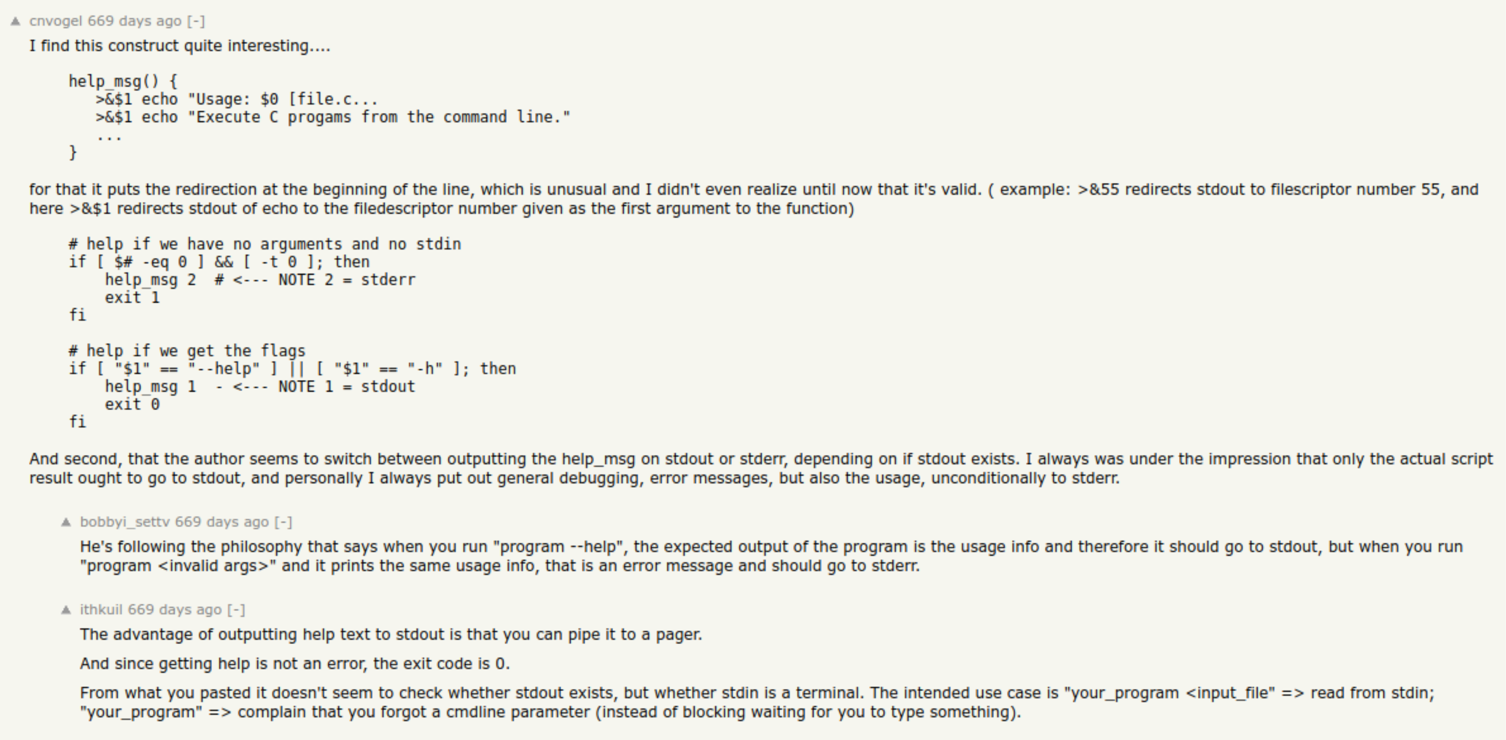
Matt Lebrun @cr8ivecodesmith · 5 Mar 2015
Hey this looks cool... Code-Compile-Run C/C++ no more. Just do Code-Run with this github.com/ryanmjacobs/c

And I also sparked quite a large discussion chain on Hacker News: (https://news.ycombinator.com/item?id=9144467)



▲ ralmeida4381 669 days ago [-]

Interesting... C in your shell is like having a sonic screwdriver in your pocket.



▲ cnvogel 669 days ago [-]

I find this construct quite interesting....

```
    help_msg() {
       >&$1 echo "Usage: $0 [file.c...
       >&$1 echo "Execute C progams from the command line."
       ...
    }
```

for that it puts the redirection at the beginning of the line, which is unusual and I didn't even realize until now that it's valid. (example: >&55 redirects stdout to filescriptor number 55, and here >&$1 redirects stdout of echo to the filedescriptor number given as the first argument to the function)

```
    # help if we have no arguments and no stdin
    if [ $# -eq 0 ] && [ -t 0 ]; then
        help_msg 2  # <--- NOTE 2 = stderr
        exit 1
    fi

    # help if we get the flags
    if [ "$1" == "--help" ] || [ "$1" == "-h" ]; then
        help_msg 1  - <--- NOTE 1 = stdout
        exit 0
    fi
```

And second, that the author seems to switch between outputting the help_msg on stdout or stderr, depending on if stdout exists. I always was under the impression that only the actual script result ought to go to stdout, and personally I always put out general debugging, error messages, but also the usage, unconditionally to stderr.

▲ bobbyi_settv 669 days ago [-]

He's following the philosophy that says when you run "program --help", the expected output of the program is the usage info and therefore it should go to stdout, but when you run "program <invalid args>" and it prints the same usage info, that is an error message and should go to stderr.

▲ ithkuil 669 days ago [-]

The advantage of outputting help text to stdout is that you can pipe it to a pager.

And since getting help is not an error, the exit code is 0.

From what you pasted it doesn't seem to check whether stdout exists, but whether stdin is a terminal. The intended use case is "your_program <input_file" => read from stdin; "your_program" => complain that you forgot a cmdline parameter (instead of blocking waiting for you to type something).