Is Linux kernel (or other low level stuff) a "good" example of how to write C [closed]

Asked 15 years, 11 months ago Modified 14 years, 5 months ago

Viewed 4k times

5

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 11 years ago.

Improve this question

Following the "what's the best use for C" question.

The Linux kernel seems to be a famous and very well thought of C program. But is it a good example of mainstream "best-practice" C?

c linux-kernel

Share edited Jan 14, 2009 at 4:00

Improve this question

Follow

Norman Ramsey
202k ● 62 ● 371 ● 541

asked Jan 14, 2009 at 0:51

interstar
27.1k ● 38 ● 118 ● 186

9 Answers

Sorted by: Highest score (default)

20

I'm prepared to get dinged for this, but the Linux kernel is a student project that got out of hand. More seriously, what the Linux kernel does well from a programming point of view is lay down a very rigid set of design and coding guidelines that make it possible for a very large number of people to contribute while still having everything fit together.

People I trust who have made operating systems their life's work tell me that the BSD kernel is a *much* better example of good C programming and good OS design. I know enough of the professors at Berkeley that I'm not surprised by this claim.

More broadly, OS kernels are highly specialized things---there's nothing *mainstream* about a kernel. If you want examples of good C practice you might try Kernighan and Pike's book on The Practice of Programming. Or if you want to study a real system, the implementation of Lua is small enough that you can read it all, extremely well engineered, and *incredibly* portable---runs on any platform with an ANSI C compiler.

Apologies for the subjective and argumentative reply :(

Share Improve this answer Follow

answered Jan 14, 2009 at 3:59

Norman Ramsey
202k ● 62 ● 371 ● 541

1 I would not find your suggestion to be surprising at all. I think that Linux's success is more sociological than technical.
– tsimon Jan 16, 2009 at 4:51

16

What do you mean by good example? I believe the Linux kernel is a very well written kernel in C with some problems, many of which are being cleaned up. However, as far as using it as an example of how to write general C code, I don't think it quite fits. Reading and understanding the kernel will teach you a lot about C. But what you won't find in the kernel is any libraries to speak of. The standard libraries are unavailable. So, for instance, you will see "printk" instead of "printf" and so forth. There is also some dark magic. The kernel is a very special case. I would not use it to find best practices, say, if you were writing some general application like a file manager or a database.

Share Improve this answer Follow

edited Jan 14, 2009 at 1:00
jfs
413k ● 205 ● 1k ● 1.7k

answered Jan 14, 2009 at 0:56
BobbyShaftoe
28.5k ● 7 ● 54 ● 74

9

Yes, it is.

If you don't want to get overwhelmed with the details of the kernel, you can start with some of the Linux utility functions written in C. See the answers to this question for details on how to get the source.

Another good place to read C source code is in C's very own Standard Library.

Finally, one more recent C project that I'm hearing good things about is [Git](https://git-scm.com/).

Share Improve this answer Follow

edited May 23, 2017 at 11:46

Community Bot
1 ● 1

answered Jan 14, 2009 at 0:55

Bill the Lizard
405k ● 211 ● 572 ● 889

Standard C Library seems to be a good starting point to me too – Gustavo Rubio Jan 14, 2009 at 1:06

1 Plauger wrote a book with really good examples from the C standard library too. amazon.com/Standard-C-Library-P-J-Plauger/dp/0131315099 – Bill the Lizard Jan 14, 2009 at 1:09

3

I would suggest looking at the gnu utilities (e.g. the source code for standard unix utilities such as 'ls' and 'awk') as they are going to be easier to understand and more likely going to be written in a more clean manner (the kernel has different performance needs and compatibility concerns than other types of software).

Share Improve this answer Follow

answered Jan 14, 2009 at 3:37

tsimon
8,010 ● 2 ● 33 ● 44

For mainstream programs? No. *nix kernel code is generally not in userland, and as such has different

3

conventions and requirements than most "mainstream" programming. It is, however, some of the best and highest-quality C code you can study to learn from. It has its weaknesses, and it's not entirely well-written, but for the most part it is very effective, clean code.

Share Improve this answer Follow

answered Jan 14, 2009 at 9:37

community wiki
Max

1

If you are asking from the perspective of learning C, then it might be too overwhelming for a start. As far as usage of C is concerned, it is best suited for applications requiring portability and performance. It is commonly used for system level programming. But, it does so at the expense of abstraction. Therefore, you have to do the bulk of coding yourself unlike a language like Java or even C++. Ideally, your application type should decide the choice of language, and with operating systems C is the way to go.

One way to go about learning about the Linux kernel is starting from a very old linux kernel version. This would contain the meat and hide the additional complexity present in the current kernel. This should be sufficient in understanding the basics.

Share Improve this answer

edited Jan 14, 2009 at 1:12

Follow

answered Jan 14, 2009 at 0:57

Epitaph
3,148 ● 10 ● 35 ● 44

I think the question was more about best practices for *using* C. – BobbyShaftoe Jan 14, 2009 at 0:59

1

Whether it is "best practice" depends on who you are asking. There are a lot of low level C programs written on Windows platform too, but if you take a look at them (e.g., device driver examples from WINDDK), you will realize that Microsoft has successfully *invented* another language that they call C :-) But still, those should probably be considered as "best practice" if you ever want to develop device drivers on Windows.

Share Improve this answer

Follow

answered Jan 14, 2009 at 1:19

PolyThinker
5,218 ● 23 ● 22

0

One excellent place is to look at the code of some open source compiler in use. You need excellent data structure and pointer skills to write a compiler.

Share Improve this answer

Follow

answered Jan 14, 2009 at 3:43

Perpetualcoder
13.6k ● 9 ● 66 ● 101

Sure, as long as you ignore GCC. Which isn't particularly great. – Cromulent Jul 11, 2010 at 21:17

0

I find the kernel to be a very good example on how to write modularized code. Despite being C, it is somewhat object oriented.

Share Improve this answer Follow

answered Jul 8, 2010 at 16:53

Maister
5,044 ● 1 ● 33 ● 34

Although the way it achieves 'object-oriented' is overwhelmingly complex, and definitely not a place to learn from for starters. I'm guessing a compiler would be a much better place to learn 'object-oriented' in C. – Vanwaril Jul 21, 2010 at 14:55