Roadblocks in creating a custom operating system [closed]

Asked 16 years ago Modified 13 years, 4 months ago Viewed 7k times

19

As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, visit the help center for guidance.

Closed 11 years ago.

It seems to me that the most common overly ambitious project that programmers (esp. Comp. Sci. grads) try to tackle is building your own operating system. (Trying to create your own programming language + compiler is probably even more common but not nearly as ambitious.)

For those (like myself) foolish enough to try: aside from the sheer size, **what are the biggest *gotcha*s or unexpected roadblocks you've encountered in trying to create your own OS from the ground up?**

Edit: A great OS question: What are some resources for getting started in operating system development?

operating-system osdev

Share

Improve this question

Follow

edited May 23, 2017 at 12:02

Community Bot

1 ● 1

asked Dec 4, 2008 at 14:02

Dinah

54k ● 30 ● 134 ● 149

It used to be creating your own text editor ranked pretty high, too. :) – unwind Dec 4, 2008 at 14:04

3 I would think biggest roadblocks are fear and doubt If you can get rid of them ... – THEn Dec 12, 2010 at 21:26

6 Answers

Sorted by: Highest score (default)

21

Being able to speak from actual experience (AROS, the biggest obstacles are:

- Hen and egg (no os <-> no apps <-> no users <-> can't attract developers)
- Why even try to compete against Windows? Apple doesn't succeed and they have lots of money and Linux doesn't succeed despite having an enormous and enthusiastic community.
- Big companies aren't interested at all to bring their apps to your OS (Adobe Acrobat Reader, MS Office,

Macromedia Flash, Java from Sun). Without these, and no good OSS alternatives, you can't attract users.

- It takes very long. In my case, it's been roughly 15 years to get a 1.0 (and we're not 100% there, yet).
- Compatibility. In order to get any users, you must be compatible to something that exists (so people can continue to use their data, etc). If you're compatible, why change to *your* OS?

So if you plan to write your own OS, you should consider this:

- It will take a long time
- For the longest time, you will be alone. I was lucky because I had this enormously fanatic and dedicated Amiga community which just wouldn't let up.
- You must find a niche where you can offer a service which no other OS can offer.
- People who can do this often get better job offers because the perception is "writing OSs is hard" :)
- You will be a member of a very small, elite group of programmers that can say "I wrote my own OS and it can do more than print 'Hello world!'"

answered Dec 4, 2008 at 14:40

Aaron Digulla

328k ● 110 ● 622 ● 834

3 "Apple doesn't succeed"? They make billions of dollars a year. That sounds pretty successful to me. – Bryan Oakley Oct 11, 2009 at 18:03

3 They just got 10% market share (tuaw.com/2009/01/02/…). That is a lot of money but they surely don't take over. So yes, in their niche, they are successful. Overall, they just can't bother the Windows dinosaur. We'll have to wait until it collapses under its own weight ;) – Aaron Digulla Oct 13, 2009 at 8:26

Any specific technical challenges besides mere size? – mudgen Mar 27, 2010 at 0:47

4 @mudge: Writing drivers for the OS is hard: It's hard to get the specs, you need to buy the hardware to test the driver, writing drivers is notoriously hard (for example debugging interrupts) and you need quite a few of them. – Aaron Digulla Mar 30, 2010 at 8:04

1 @DorkFace Billions of dollars make up the struts which keep it up... – Aaron Digulla Sep 4, 2017 at 9:18

9

Been there, done that. The biggest obstacle, for me at least, was device drivers. Coding the OS core is the "fun part", however it is useless without being able to do I/O (disk, keyboard, video, network, at least). Today, if I again had the time and will to engage into such project, I'd probably target Xen VM instead of the raw hardware -- if for nothing else, then because it hides many ugly idiosyncrasies of x86 as well as hardware. Xen presents a nice uniform, hardware-independent view of I/O devices, and still gives you enough freedom to play with

the "interesting" OS parts (mm, process management, synchronization, interrupts, etc).

Share Improve this answer Follow

answered Dec 21, 2008 at 20:59

zvrba

24.5k ● 3 ● 56 ● 65

yes I am stuck at this point even it is my first :c – u185619 Sep 29, 2015 at 16:04

5

The biggest Roadblock? I think it is when you find out how much gaps there are in the specifications and how many bugs there are in the various implementation.

Seriously, even if you would have all the Specs for all the components (i.e. USB, DMA, IRQ, your CPU...) you will find that a) some things are not specified (i.e. what happens if you send a certain sequence to your USB Device) and b) some things are just bugged and you need to work around them (i.e. the dozens of Bugs in the CPU that are detailed in the CPU Errata that both Intel and AMD publish)

I don't know how many workarounds for bugs there in a modern operating system, but since Linux and *BSD is open source, their drivers tell you much, i.e. this one. And expect to get some seriously negative side-effects like this one.

So yeah, If you try to write an OS, be prepared to curse a lot at hardware manufacturers and start to lose your faith

in the quality of modern PCs :-)

Share Improve this answer

Follow

answered Dec 5, 2008 at 18:48

Michael Stum

181k ● 119 ● 407 ● 540

> "be prepared to ... start to lose your faith in the quality of modern PCs" No doubt! This reminds me of something I read once to the effect of: if you don't want to see a hack, don't dig through the code. Ie: once you know the deeper internals, you'll see how shoddily it's all put together. – Dinah Dec 5, 2008 at 18:58

3

I think nowadays a lot of people who are otherwise good programmers don't know how little they know about how computers work. You'll need a serious grip on the fundamentals to get a machine to boot your OS from a disk, and that type of knowledge is spread pretty thinly nowadays. You won't find a shelf of books about it nowadays, either.

Interestingly, I asked a question on SO a few weeks ago which involved the type of knowledge which was the bread-and-butter of serious PC programming 10-15 years ago, and one commenter said they considered it to be a *hardware* question.

I'm not knocking them at all, but I thought it was an interesting reflection on how skills have changed.

Share Improve this answer

answered Dec 5, 2008 at 18:38

Follow

Will Dean
39.5k • 11 • 92 • 118

2

I think by far the single biggest roadblock is users - until you've built the majority of a working system, you're highly unlikely to have any users. Without users, you don't have feature requests, bug reports, and consequently there's limited motivation. However, even if you do build a working OS, there's no guarantee you'll get a user base unless you can find a hook that draws people in - what sets your OS apart enough for people to be willing to try beta versions, hang in there when there's hardly any working applications ported to it, etc.

You might be interested in checking out SkyOS - it was developed largely by a single developer (now a small development team). The Wikipedia article has a good summary as well. I think it's a good example of how such a daunting task can be undertaken and turn out pretty well. They've done some interesting things with SkyOS, and it's especially cool to see that come in large part from the efforts of a single person in their spare time.

Share Improve this answer

Follow

answered Jan 18, 2009 at 21:12

Jay
42.6k • 14 • 68 • 83

Writing an OS encounters the same problems as other large software projects.

0

It lacks clearly defined goals, and has timeline / estimation problems due to lack of experience.

Also it would be useful for the programmer first to create a 'hello world' type operating system, so that he/she learns the concepts behind OS development, and then can focus on developing an OS (not learning OS concepts).

Share Improve this answer

Follow

answered Dec 5, 2008 at 15:46

eric

87 • 1

I think is harder, there are no libraries functions etc. You need to make everything from scratch. – skywalker May 1, 2016 at 16:29