Eudyptula Challenge

`eudyptula-challenge.org`

Google “linux kernel challenge”

LinuxCon Chicago, 2014

6847 people
No advertising

What

Why

Numbers

Hints

“I find the Eudylipta challenge to be a well paced foray into kernel alterations, which has kept me interested in an area of computing I would have never otherwise considered. I find Little to be endearing, and am always excited to finish a level.”

– Emily Saunders Walmsley

The Eudyptula Challenge

What is it?

The Eudyptula Challenge is a series of programming exercises for the Linux kernel, that start from a very basic "Hello world" kernel module, moving on up in complexity to getting patches accepted into the main Linux kernel source tree.

How do I do it?

Email `little at eudyptula-challenge.org` and say that you want to join in. You will receive an email with your first exercise and details on how the challenge is run. All communication goes through email, just like the Linux kernel is developed.

Please note, all HTML-formatted email will be merrily rejected, please fix your email client to not send HTML email if you wish to do this challenge. Linux kernel mailing lists reject HTML email and so do we.

What are the rules?

There aren't any, with the exception that you really should be doing your own work. This challenge is for you, not anyone else, so don't ask about how to solve them on mailing lists or IRC. Posting answers to the questions online is also frowned apon, as it really doesn't help anyone out.

Also, we can tell if you are cheating, we have our ways...

What skills do I need to take this?

A basic understanding of the C programming language is required. No previous Linux kernel coding experience is needed, but that of course will help.

Where did this idea come from?

The idea for this came to us after a long night of drinking in which it was determined that if the Linux kernel was to survive, it would need new programmers to fix all of the bugs that were recently added after a long night of drinking.

This Challenge is also modeled after the wonderful Matasano Crypto Challenge, which, if you haven't taken it

20 tasks

- "Hello World"
- Built custom kernel from git
- Coding style
- Submit patch to mainline
- And then it gets hard...

Biggest complaint

“The bad thing is that the response times are very lengthy.”

“feedback can take a while to come”

“Is my task lost?”

Biggest complaint

“I love how you've emulated the typical process of putting in a patch and the many things one has to know as a kernel dev.”

— Gideon

Why?

“I like the fact that no solutions are provided for the tasks, because that's largely what the process of writing software is like – you have to accomplish something, and you're not sure how to do it. And once you complete a task, you're not entirely sure that you've found the "best" solution. All too often students expect to eventually be provided with a solution, which isn't necessarily the way life works. I think the Eudyptula Challenge helps students develop their research muscles, which makes them better programmers.”

–Scott Edwards, Professor of Computer Science

How Many?

6847 people

3801 people never submitted task 01

3046 “active”
1134 on task 01

1444 people have built a custom kernel from git

“Getting a custom kernel built was harder than I remember from 2.2 days, but now I've got it installed my firewall is running about 8°C cooler than it was last week.”

—Orsic Wilkinson

172 people have
a patch accepted
into the kernel tree.

“Linux staging patch makes a kernel programmer not. Yeesssssss.
Old and moldy, the source is. Hmmm.”

– Michael

“I think the challenge is just a strategy to clean up the mess in the staging directory…”

46 people have completed the challenge

1 person got a job

“I'm a student and have had a job that payed my bills but that I wasn't excited about. I, just now, got a job as an embedded software engineer because of this challenge. The interview was focused around it :) It's one of the best things I've done during my free time. true story.”

– Martin Kepplinger

Hints

Don't use gmail

Don't use Apple Mail

Don't use Outlook

Don't send .ko files

Don't send tarballs

Do your own work

Don't ask for help in public

Don't put code on github

Ask Little for help

Have fun

“Super. Challenging and encouraging, despairing/frustrating sometimes, but always fun.”

– Fernando

Photograph by Paul Souders | National Geographic Photo Contest 2013

© Copyright Paul Souders. All rights reserved.

Eudyptula Challenge

`eudyptula-challenge.org`

Google “linux kernel challenge”

LinuxCon Chicago, 2014

No one can pronounce it or spell it, just Google “linux kernel challenge” to find it.

This talk was presented at the LinuxCon North America Conference in Chicago on August 22, 2014.

As of today (08-20-2014)

No “advertising”, one email to kernelnewbies, and one Google+ post.

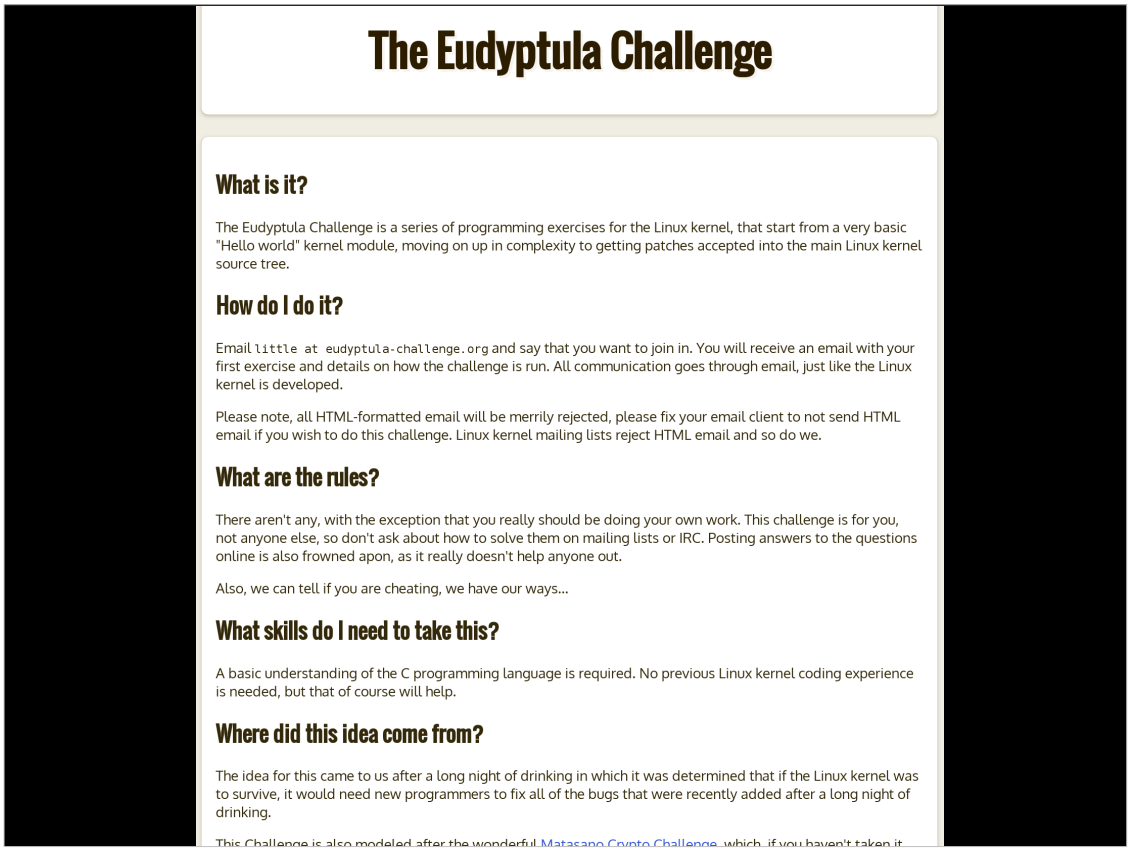
What
Why
Numbers
Hints

I'm going to cover What the challenge is, why it is set up the way it is, how it works, and how many people are doing it and other fun numbers.

And then provide some hints for people taking the challenge for how to make things better for themselves.

> **“I find the Eudylipta challenge to be a well paced foray into kernel alterations, which has kept me interested in an area of computing I would have never otherwise considered. I find Little to be endearing, and am always excited to finish a level.”**
>
> **– Emily Saunders Walmsley**

And I'm going to give real quotes from people who have taken the challenge, or are in the middle of taking it.



The Eudyptula Challenge

What is it?

The Eudyptula Challenge is a series of programming exercises for the Linux kernel, that start from a very basic "Hello world" kernel module, moving on up in complexity to getting patches accepted into the main Linux kernel source tree.

How do I do it?

Email `little at eudyptula-challenge.org` and say that you want to join in. You will receive an email with your first exercise and details on how the challenge is run. All communication goes through email, just like the Linux kernel is developed.

Please note, all HTML-formatted email will be merrily rejected, please fix your email client to not send HTML email if you wish to do this challenge. Linux kernel mailing lists reject HTML email and so do we.

What are the rules?

There aren't any, with the exception that you really should be doing your own work. This challenge is for you, not anyone else, so don't ask about how to solve them on mailing lists or IRC. Posting answers to the questions online is also frowned apon, as it really doesn't help anyone out.

Also, we can tell if you are cheating, we have our ways...

What skills do I need to take this?

A basic understanding of the C programming language is required. No previous Linux kernel coding experience is needed, but that of course will help.

Where did this idea come from?

The idea for this came to us after a long night of drinking in which it was determined that if the Linux kernel was to survive, it would need new programmers to fix all of the bugs that were recently added after a long night of drinking.

This Challenge is also modeled after the wonderful Matasano Crypto Challenge, which, if you haven't taken it

The Eudyptula Challenge is a series of programming exercises for the Linux kernel, that start from a very basic "Hello world" kernel module, moving on up in complexity to getting patches accepted into the main Linux kernel source tree.

Everything is done through email.

Modeled after the Matasano Crypto Challenge, R.I.P. :(

20 tasks

- “Hello World”
- Built custom kernel from git
- Coding style
- Submit patch to mainline
- And then it gets hard...

There are a total of 20 tasks in the challenge.

Things start out easy with a simple “Hello World” kernel module, then building a custom kernel from Linus's git tree, figuring out module loading, coding style issues, some other basics of drivers, submit a patch to the mainline kernel.

Then the shit gets real.

Biggest complaint

“The bad thing is that the response times are very lengthy.”

“feedback can take a while to come”

“Is my task lost?”

Yes, it's slow.

It was never expected to take off like it did.

The first few months, response time was at most 2-3 days. Now it can be 2-3 weeks, some task queues are a month long.

Biggest complaint

“I love how you've emulated the typical process of putting in a patch and the many things one has to know as a kernel dev.”

– Gideon

But, this is how the real world works. You submit a patch and then wait an unknown amount of time. There are no deadlines in the kernel developer community that must be met. Patches are reviewed when they are reviewed. Sometimes you need to remind people to review them. Sometimes you need to resubmit.

Patcience is a virtue.

Relax

Breath

Chill out with your bad self.

It's all about how people learn.

No matter how much you tell someone what they should do, only if they have to do it themselves will they actually understand it.

In education, tests try to simulate this. If you force people to dig for an answer, they know it better than if they were told it.

No matter how much you tell someone something, they usually don't believe you, or really understand, until they do it themselves.

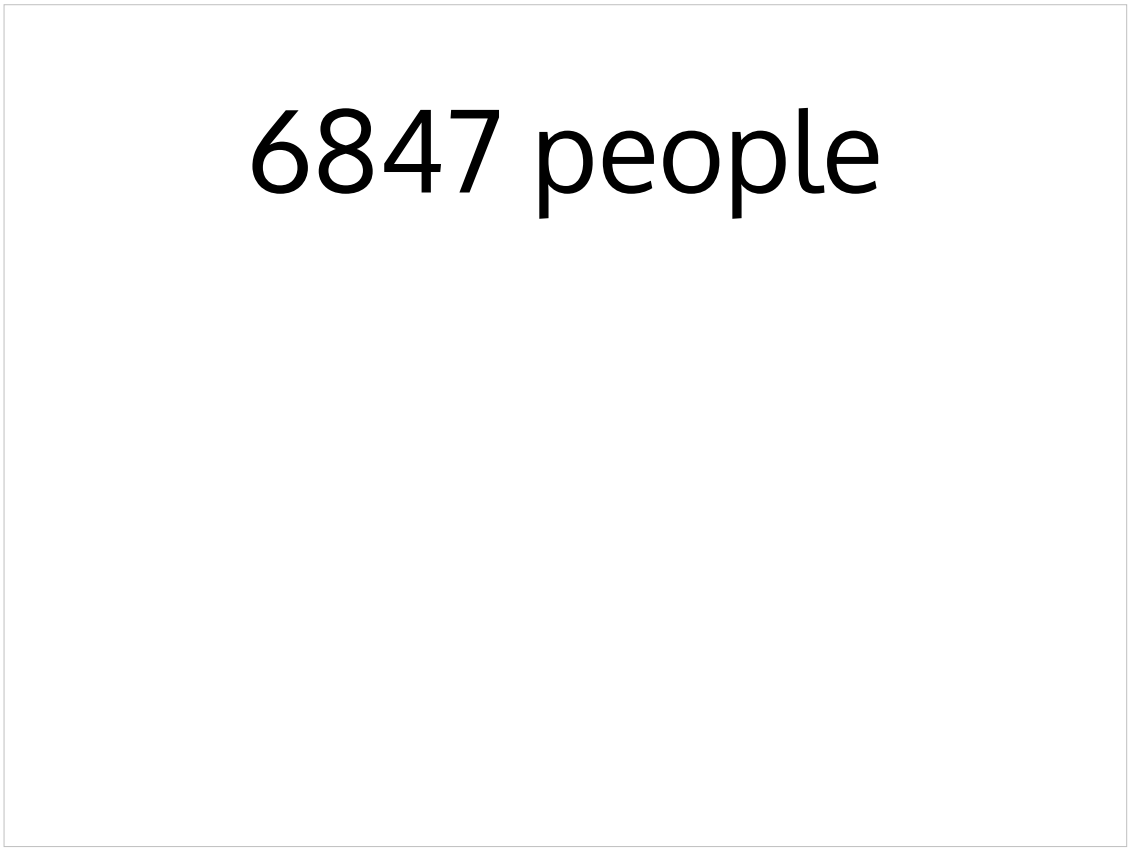
“I like the fact that no solutions are provided for the tasks, because that's largely what the process of writing software is like – you have to accomplish something, and you're not sure how to do it. And once you complete a task, you're not entirely sure that you've found the "best" solution. All too often students expect to eventually be provided with a solution, which isn't necessarily the way life works. I think the Eudyptula Challenge helps students develop their research muscles, which makes them better programmers.”

–Scott Edwards, Professor of Computer Science

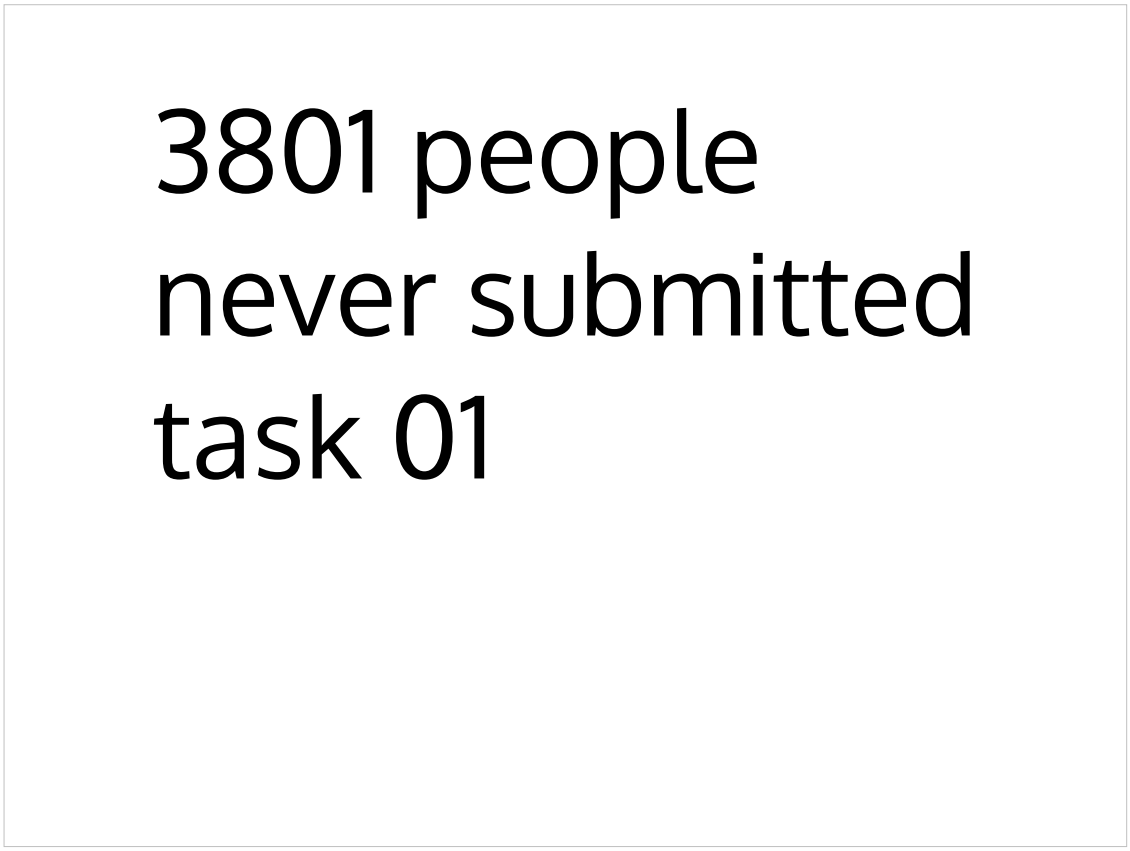
Look, a real professor took the challenge and agreed that this is a real-world simulation.

How Many?

Let's talk numbers, everyone wants to know how many people are actually doing this challenge, and how many are finishing it.

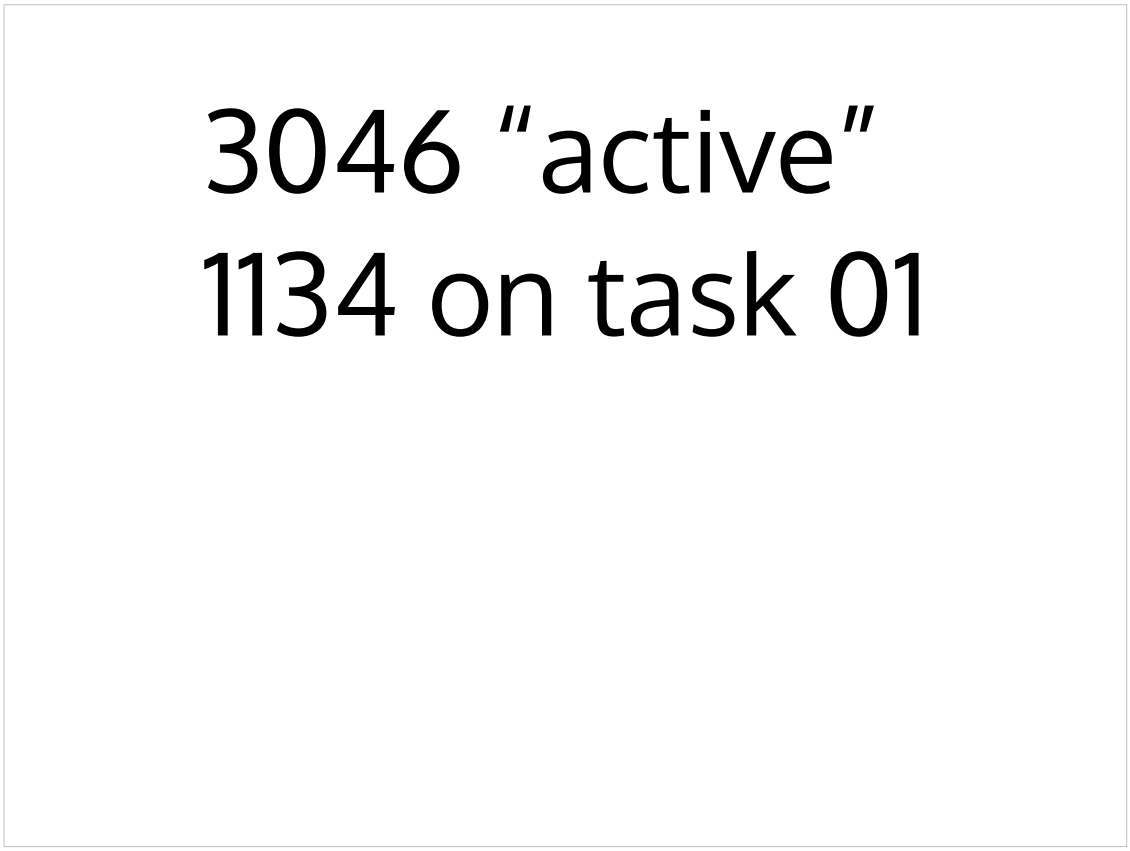


As of today (08-20-2014)



So that's 55% dropout, not bad.

People get removed if you never respond to the first task, and it is over a month from signing up to try to track “active” users.



So that gives us 3046 “active” people.

Right now, 1134 are still on task 01

That means they have submitted at least one response to task 01, or they have signed up in the past month and haven't been “cleaned up” from the system.

1444 people have built a custom kernel from git

That's more people than attended this conference.

> **“Getting a custom kernel built was harder than I remember from 2.2 days, but now I've got it installed my firewall is running about 8°C cooler than it was last week.”**
>
> **–Orsic Wilkinson**

Look, we are saving people power!

Lots of people have responded that they really liked just building their own kernel as it gave the the feeling of control over their machine that they didn't have before.

172 people have a patch accepted into the kernel tree.

Yes, some already were kernel developers that had submitted patches before, but that was just a handful. Almost all of these are “new” developers. Some have gone on to submit, and get accepted, many patches.

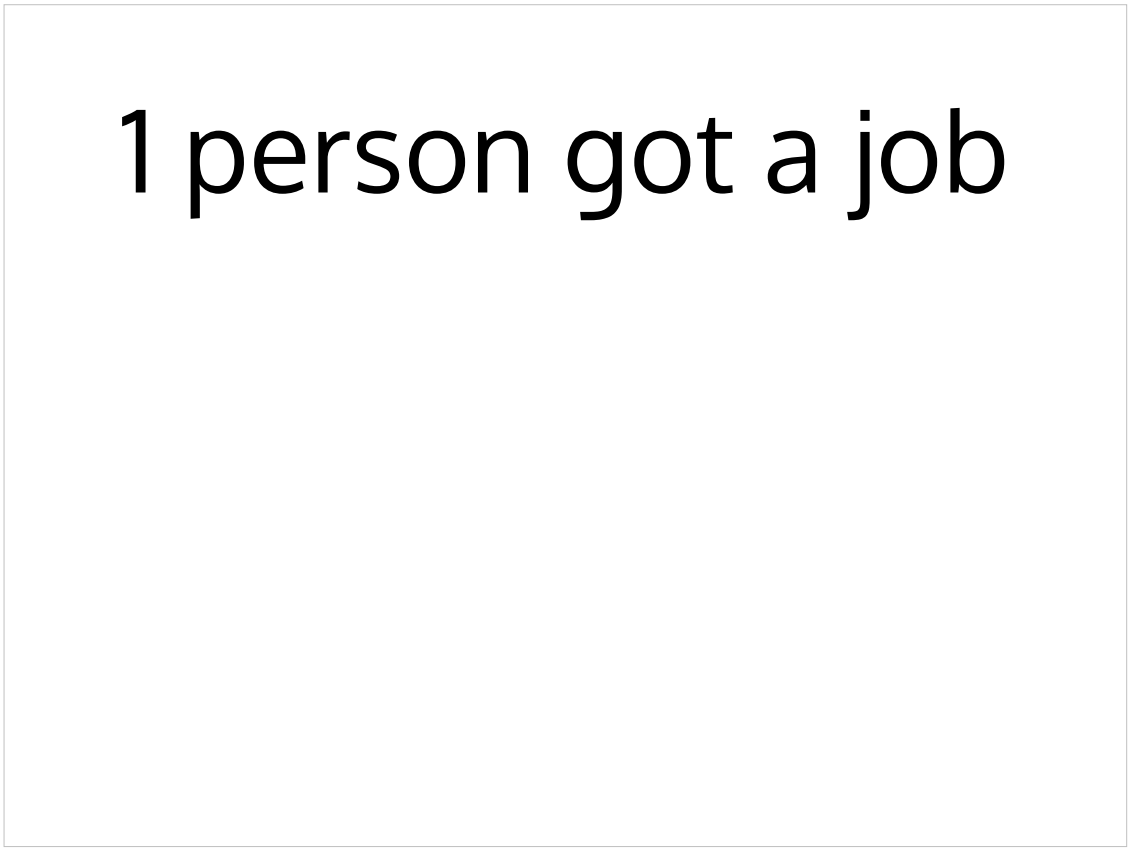
“Linux staging patch makes a kernel programmer not. Yeesssssss.
Old and moldy, the source is. Hmmm.”
– Michael

“I think the challenge is just a strategy to clean up the mess in the staging directory...”

Yes, a patch in the staging tree doesn't make you all of a sudden a “real” kernel programmer. But lots of people have started out that way, one was up on stage yesterday at the keynote, so don't think it isn't a gateway drug, it is.

.6% finish this thing. It's not easy.

Every single one of these people want there to be a second round of tasks. It is being worked on.



If this challenge stops tomorrow, I am happy, this is way more than was ever expected.

“I'm a student and have had a job that payed my bills but that I wasn't excited about. I, just now, got a job as an embedded software engineer because of this challenge. The interview was focused around it :) It's one of the best things I've done during my free time. true story.”

– Martin Kepplinger

Hints

Ok, now that you want to do this, let's give some hints on how to do this properly so you don't fall into the very common traps that people have in the past.

Don't use gmail

Gmail defaults to HTML email, which automatically gets rejected by the system.

Even if you figure out how to turn that off, which takes some time, gmail messes up attachments by using base64, which can not be turned off.

If you are stuck with gmail, use it as an IMAP server, and use mutt / thunderbird / evolution / alpine / something real instead.

Don't use Apple Mail

Don't laugh, lots of people do the challenge on OS-X in a virtual machine, which is fine.

But then they submit the tasks using Apple Mail, which does unspeakable things to attachments, so much so that they are unreadable.

No kernel patch can ever be applied from Apple Mail, so don't even try it, just use your Linux virtual machine and an email client in it.

Don't use Outlook

Again, does horrible things to email attachments, making them impossible to read.

The first task tests all of this, which is what stops a lot of people cold. Configuring an email client to work properly, so that it sends patches in a format that is not corrupted is a non-trivial task.

See the kernel file, Documentation/email_client.txt for how to do this properly.

Don't send .ko files

Don't send pre-built binaries, they are useless, the system will just delete them.

The source code is what is needed, not anything already built, as your machine is different from the virtual machine the task is running in, they are guaranteed to not work.

Don't send tarballs

Or zipped files, or .bzip or .gz or anything compressed in any form.

Send text, that's all.

That's how the kernel developers work, it's stable, reliable, and “just works.”

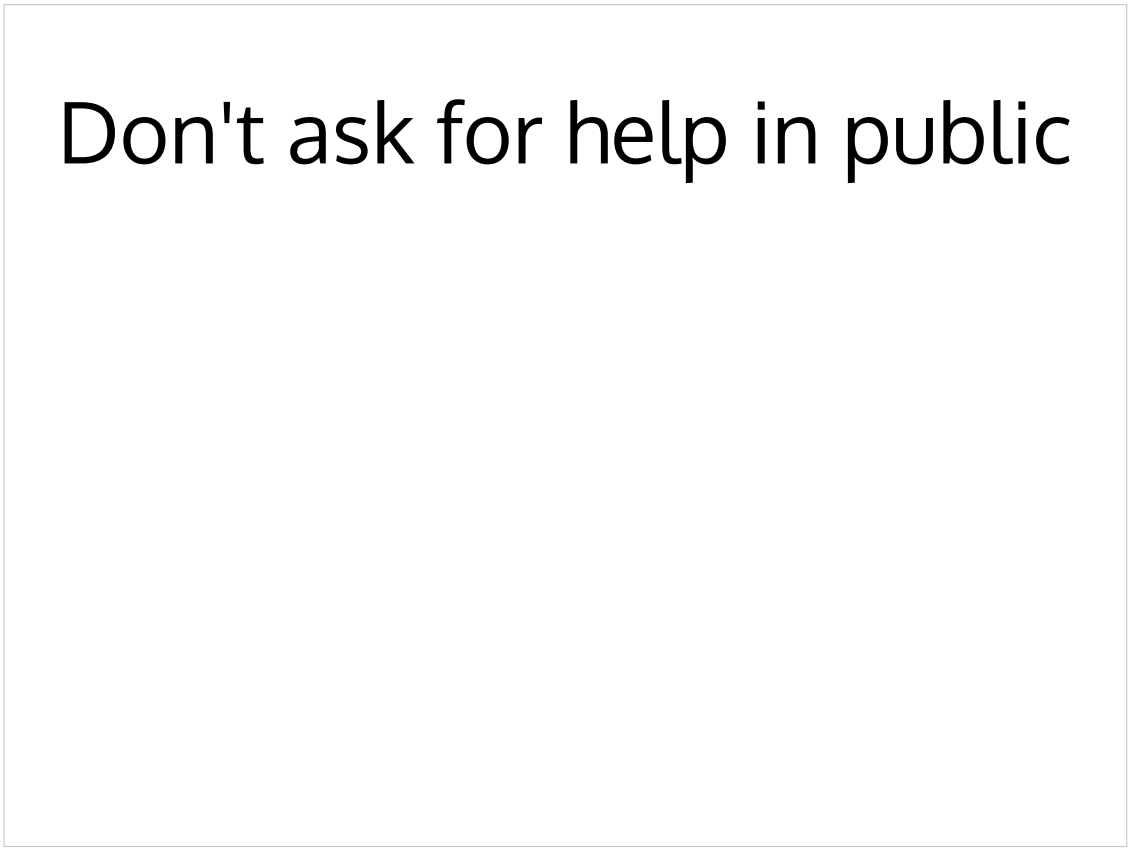
Do your own work

The challenge is for you to learn, not for you to copy from someone else.

It's fine to use examples from others, and expected, there's lots of code out there to build on top of, that is fine. But if you do, you HAVE TO properly attribute it.

If not, and you are found copying without attribution, you will be removed from the challenge.

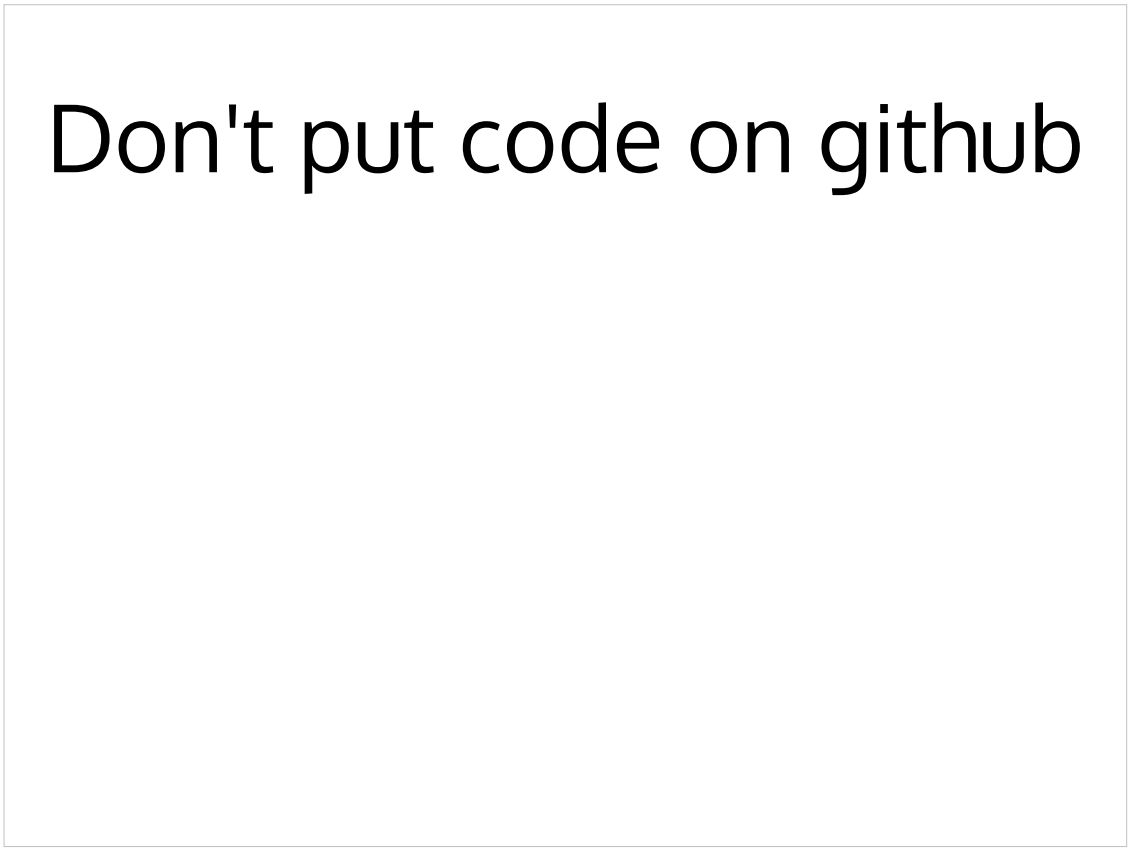
Don't ask for help in public



The challenge is for you to learn, not for you to spoil for someone else.

No one outside of the challenge cares about the tasks it is having you do, don't bother them.

If you ask for help in public on a mailing list, or IRC, or Stack Exchange, or anywhere else, and it is found, you will be removed from the challenge.



Or any other public site (like a blog or G+ or Facebook.)

If it is found, you will be removed from the challenge.

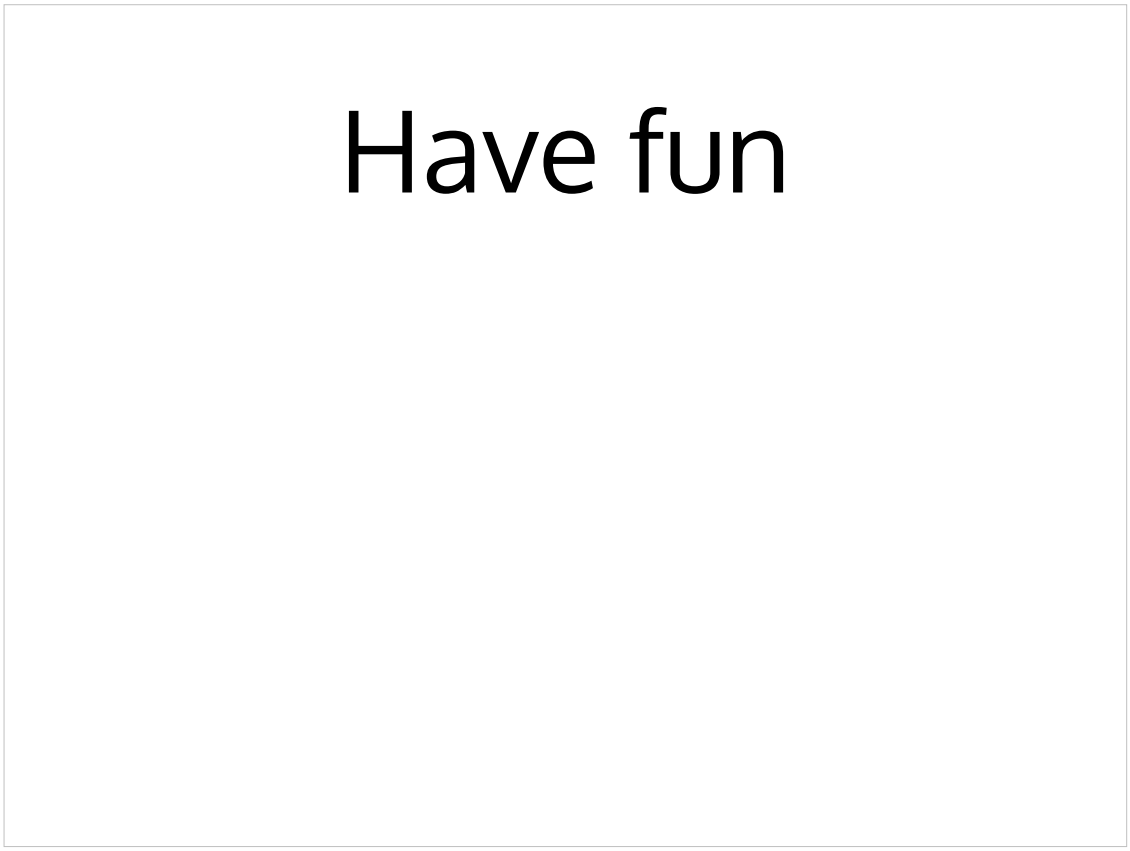
This has happened a number of times already, and the people were removed, don't let it happen to you.

Ask Little for help

But, if you have a question, and are stuck on something, or are unsure about what a response to a task means, just ask. The system will flag it and will respond with either a hint, or a better explaination, or something else.

It's fine to ask for help about the challenge from the challenge itself, but be prepared to show what you have done to try to figure out the answer already.

This is not a tutorial, it is a challenge, answers will not be provided, but hints on how to find the answers will be.



Most of all, just have fun with the challenge. It's meant for people to enjoy, not stress over.

“Super. Challenging and encouraging, despairing/frustrating sometimes, but always fun.”

– Fernando


NATIONAL GEOGRAPHIC
Photograph by Paul Souders | National Geographic Photo Contest 2013
© Copyright Paul Souders. All rights reserved.