# How to write a good Google Summer of Code Proposals

W. Adam Koszek

*Koszek ORG*

wojciech@koszek.com

2022-10-05T17:46:51Z

Let me keep it short. As short and concrete as your proposal should be. I document my examples with this years proposals, but the fact you see title of your project here means nothing. These are just examples.

### Project title

Nothing more here. Keep it short and precise. ***NTFS for FreeBSD*** is so much different than ***Proposal for FreeBSD***. Positively different.

### Short description

I always try to think that **lesser is more**. Please keep it short. Very short. Lets make it 2-sentence short. So short description can contain 1 sentence answer for each of the questions:

- **What this proposal is all about?** Example: *I propose to improve FreeBSD support of filesystem X*; *I propose to improve 3 FreeBSD text utilities: grep(1), diff(1) and sed(1)*; *This proposal is about bringing support for CPU percentage limits".
- **Why do you work on this project?** Example: "I want to improve support for colorful ls(1) because current ls(1) is archaic"; "I want to improve FreeBSD ULE kernel scheduler to let me turn CPU off"* etc..

Most of the short descriptions of the projects were surprisingly good. ### Provide real name and surname

If possible, provided something that can be entered with limited set of characters. Basically something that consists of a-zA-Z characters. The reason is that we all have limited knowledge of the world and to save ourselves lots of confusion, lets keep things simple. **Please provide name you'll be using over duration**

**of the project** If for some reason name from the Google form is different that the one which you use on a daily basis, please point it out in the proposal right away. This saves us some time too.

### E-mail, phone, IRC, language

Nothing special goes here. Whichever e-mail is fine, as long as you check it on regular basic. Phone . . . probably won't be used, so I wouldn't be too much afraid.

IRC is very useful – please mention precisely what's your typical nick when where are you typically hanging out. FreeBSD has a dedicated IRC channel:

#freebsd-soc

You can meet actual mentors there. Once you join, you can feel free to ask questions about GSOC.

### Availability

Provide relatively accurate availability: number of hours per week which you can spend on the project. **Be honest**. We all know 8 hours per day of actual coding is rare if you have school, assignments and real life. Yes, doing what you want (FreeBSD is what you want, right?) makes GSOC fun and you can spend more time on it, but don't make this number higher artificially.

So here is important thing: **I don't know what's the Google's expectation** for these numbers. If they expect "8 hours" . . . . well . . . you have to put 8 hours. Actually 8 might be right.

(I think the secret idea is to figure out which people can handle: 4[weeks]x4[days]x8[hours] schedule and steal them to Googleplex later)

Remember what those 160 hours mean: it means hours on refining the project, answering e-mails, coordination, delays, waiting for your mentor to review your work, figuring lots of obscure problems on your own, cursing BSD community (well, other communities are bad too, but these are only BSDs that are identified with Beastie).

### Bio

This is important for me. Lets assume you're applying for the FreeBSD project. Some stuff which I'd expect you to write in a bio is:

- *Where and what do you study?* No explanation needed, right?
- *Have you ever worked professionally with computers?* If you did, it'd be worth mentioning what you did in the computer field (sys-admin, programming, Web development, graphics etc. . . – everything counts)

- *Do you already have experience with FreeBSD?* Examples: **"I have no prior experience with FreeBSD"**. This is perfectly valid statement. Other possibilities: **"I work on FreeBSD in a free time for last 2 years; I have 3 PRs submitted to FreeBSD bug database**
- *Which OS do you use on a daily basis?* Examples: **"On a daily basis I use Windows, but I work on FreeBSD in VirtualBox"**, **"I work on GNU/Linux and I work remotely on FreeBSD server**. Motivation behind this one is to assess, how painful will be bringing you on track with development environment and methodology.

**Possible mentor**

This section is important to me, since it shows me you've researched the topic well. Mentioning person who I know has an expertise related with your proposal means you:

- follow the state-of-an-art work of the project, e.g.: that you follow FreeBSD mailing lists This is good: shows your involvement in the community
- know this person, since you're interested in his/her field and you've contacted them already and they'll know you, if I ask them about you.
- you know this mentor in person, since he's a part of your university staff or you're from the same country and you met him/her on the users conference.

All of it is very good, since it shows you did some additional work to actually help your proposal get accepted. **Yes, you can actively lobby to help your project**. Basically: if I see you actively pushing people over e-mail/IRC to help you with finding a mentor, this is a good sign.

If no activity is done from your side, this is bad.

IRC channels: #freebsd-soc and #bsdcode on EFnet are the good place to hunt for mentors. You can ask on a mailing list too. We have a mailing list for this purpose too. However, I don't know what its activity. So maybe ask on have a freebsd-hackers. This is probably pretty darn good way. ### Project Description

Before you jump to the project description - I sort of miss "motivation" section in this whole Google/Melange world. I think having a motivation backed by good arguments is important. So please include answers to: Why did you pick this project? Why do you think it's interesting? This will help us understand what's driving you in the GSOC train.

Anyway: I'd expect you to provide in-depth explanation of the project next. A lot of projects are very different in between each other, and it's hard to give a guideline, but just like they teach us all: introduction, content, summary. This is how I would like to see GSOC project descriptions stated. Some questions which you may find helpful: What's the project all about? What does it solve?

What current system/program doesn't have, that your project will bring? Who will benefit from it?

Don't include all answers to all questions. "Project description" must be technical, and not only having brave statements. So include technical details too:

- What's wrongly implemented/architected in the current version of a program/system. Why do you think so?
- How your project will change that? What changes you'll bring? What advantages will your new implementation have over the current state of things? What disadvantages?
- How do you plan to accomplish it? Will you concentrate of getting it all running in whichever way (prototype)? Or you'll work with a mentor to deliver great code which will be ready to be submitted to FreeBSD Subversion repository?

## Deliverables

This is pretty short, but important. "Deliverable" is what you deliver. This is more or less your schedule.

This isn't all that rocket science. I often have problems with assessing time. So just take several assumptions: you have 160 hours. Let say 60 hours are spent on: interruptions, coordination, reading manuals, documentation, web pages, blog entries etc. which help you figure out what's the stuff you're working on all about. Basically: 60 hours of cost related with learning and catching up.

This gives you 100 hours for doing real work.

Typically you'd jump and start coding. Yes, I know, I know. However, it would be good to have a sketch. It would be good to have a schedule.

Can't help you with that. Really. Joel Spolsky has something for you: Painless schedules

(By the way, he wrote tons of good stuff on his blog, which is summarized in his two books:

Really excellent resources of knowledge, I can highly recommend)

Anyway, lets get back to deliverables. Example:

- Krash_Report prototype hacked in .sh; 5-line pass/fail report sent through telnet(1) to HTTP server and displayed as pure HTML remotely" June, 7th, 2012.
- Improved Krash_Report #2 with client fetching full ASCII dump of kernel failure and uploading it over telnet(1) to HTTP server. Displayed as HTML. June, 14th, 212.

- Improved Krash_Report #3 with server side implemented in PHP and letting you to login and see the report in HTML form with CSS. June, 21st, 2012.

Note dates. Note specifics. Basically it's good to see whether you're on track or not. It helps you, it helps your mentor.

## Test Plan

If you work on FreeBSD, ask first. In FreeBSD base we have some directories that contain unit tests. Provide something that tests what you've done. Having type 17 commands is always worse than typing 2-3 commands. In the perfect case you'd provide 1 script that tests whatever you've made.

Of course that's very generalized case: if you have worked on very specific kernel-level project, you'll have to provide other means of testing framework.

However, you get the point. Provide your mentor a way to systematically test your work and actively give you feedback.

If I have to type:

```
./doit.sh
```

to observe what you've done already, it's much more likely I'll do it and provide some feedback. The end, Hope that helps