Name: Ahmed Ehab

Street Address: 28 Adnan Madany, Mohandeseen

City: Cairo

Email Address: Ahmed.bahloul00@eng-st.cu.edu.eg

Phone: +20-100856356

University: Cairo University

I am a 21-year old computer engineering student. I am currently pursuing my undergraduate studies at Computer Engineering Department of Cairo University, Egypt. I have 3.98/4 GPA and ranked first on my class for the previous three years. My topics of special interest are Operating Systems, Algorithms and Artificial Intelligence. I started out with C/C++ in my first year of undergraduate studies and moved ahead to try my hand at Javascript, Python.

Last Semester, I took Operating Systems course in university, it was a good course which included a lot of topics as Process Management Scheduling, Process Management Synchronization, Deadlock, Memory Management, IO Management in the theory point of view. While practically, we had all our labs in linux and we had made an OS project in C language.

Operating Systems Project
https://github.com/ahmedehabb/OS-Process-Scheduler

Built a CPU Scheduler which determines an order for the execution of its scheduled processes.
Implemented three algorithms: **HPF**, **SRTN**, **RR**.
The scheduler should be as much optimized as possible memory and time usage.
Stages:
Process Generator: Creates a data structure for processes and provide it with its parameters.
Clock: Clock module is to emulate an integer time clock.
Scheduler: Keeps track of the processes and their states and it decides which process will run.
Process: When a process finishes it should notify the scheduler on termination.
Memory Management: Added memory allocation capabilities using the buddy memory allocation system.
Input / Output: Includes Simulation & OS Design Evaluation.

I would gain an in-depth knowledge about Operating Systems and how to write System calls properly. Though I have done a lot of projects involving C, but none of them have taught me to write system calls. As this project would actually be a real contribution to a real operating system.

I’ve read a little bit about Adding a new system call to the Linux kernel and I tried to add a very basic system call which just prints hello world and encountered the steps which includes visiting the system call table and declaring it.

<u>Algorithm</u>

After reading articles about adding system calls I think I've got a pretty good idea the steps of adding the system calls

Steps :

1. Detecting the missing syscalls at first and determing what they do and what are the requirements of implementing this sysCall, in other words, Do they depend on each other and Do we need to do them in any order due to a timeline for example or this isn't a constraint

Let's say we know had func1() to be implemented and added to the kernel.
We must have sequence of steps now:

2. **Designing The API :** it's a very good idea to discuss the interface on the kernel, and it's important to plan for future extensions of the interface.
    - Determining the right arguments for the function
    - Future extensibility and how to handle this way. As an example for this is the more sophisticated system calls that involve a larger number of arguments, it's preferred to encapsulate the majority of the arguments into a structure that is passed in by pointer as it's much easier with Future extensibility by including size parameter for example

3. **Proposing The API :** To make new system calls easy to review, for each system call we need to include the following
    - The core implementation of the system call
    - Documentation including a demonstration of the use of the new system call

4. **SysCall Implementation**

5. **Testing**

    - A new system call should obviously be tested.
    - It is also useful to provide reviewers with a demonstration of how user space programs will use the system call.
    - A good way to combine these aims is to include a simple self-test program in a new directory under `tools/testing/selftests/`.

<u>Timeline:</u> For this experimental proposal, I think if the project is 350 hours , nearly 6 hours a day if working 5 days a week.

I really don't know the sequence needed of implementing these syscalls will be implemented simultaneously or one after another.

I hope I will get better insight after my first email.

I hope I had made a good proposal but I need to know some information if you don't mind.

My Questions:

1. **I really don't know the pattern of implementing these syscalls as I mentioned before whether they will be implemented simultaneously or one after another.**
2. **Can I know the system calls needed to be implemented to get a better insight of what I'll be doing?**
3. **I really didn't find the repo or link for the code base. So can you provide me with it or pointing to the part of the code I will be interacting with**

Obligations:

I have no obligations during the summer and would be able to contribute full time to GSoC.

References:

1. LWN article from Michael Kerrisk on use of flags argument in system calls: https://lwn.net/Articles/585415/
2. LWN article from Michael Kerrisk on how to handle unknown flags in a system call: https://lwn.net/Articles/588444/
3. LWN article from Jake Edge describing constraints on 64-bit system call arguments: https://lwn.net/Articles/311630/
4. Architecture-specific requirements for system calls are discussed in the *syscall(2)* man-page: http://man7.org/linux/man-pages/man2/syscall.2.html#NOTES
5. "How to not invent kernel interfaces", Arnd Bergmann, https://www.ukuug.org/events/linux2007/2007/papers/Bergmann.pdf