Introduction

Welcome to Project Threads! In this project, you will add features to the threading system of Pintos.

In Project User Programs, each thread that you dealt with (except the `init` and `idle` threads) was also associated with a user process, with its own address space, data backed by an executable file, and ability to execute in userspace. Importantly, each thread that corresponded to a userspace process was the only thread in that process; multithreaded user programs were not supported.

In this project, you will overcome this limitation by adding support for multithreaded user programs. Moreover, you will implement an efficient alarm clock and strict priority scheduler. However, for simplicity, you will implement these features only for kernel threads – threads that only execute in kernel mode and have no userspace component.

The details of this assignment can be found in the Tasks section. However, you may find it helpful to first read through some of the CS 162 Pintos documentation. This will likely help you understand the required tasks.

Even if you read through some of these during previous projects, we recommend (re)reading through the following sections of the Pintos documentation:

- Threads
- Synchronization
- Development

Setup

First, log into your VM. You should already have your Pintos code from Project User Programs, which you will be building off of. We recommend that you tag your final Project User Programs code since you will probably build off of it for Project File Systems.

```
cd ~/code/group
git tag proj-userprog-completed
git push group main --tags
```

If you are using the staff solution for Project User Programs (check Ed for more information), you will need to reset your repo.

```
git fetch staff main

rm -rf src/

git checkout staff/main -- src

git commit -m "Reset to skeleton code before applying diff"

git pull staff main

git push group main
```

Then, you can apply the given patch `p1.diff`. Make sure `p1.diff` is in `/code/group`.

```
patch -p4 -i p1.diff
```