CMPSC 400
Operating Systems
Spring 2020

Lab 4 Assignment:
Studying a Miniature OS
Submit deliverables through your assignment GitHib repository bearing your name. Place source code in `src/` and written work in `writing/`.

Objectives

We will studying a miniature OS tutorial for miniature OS called *BareBones*. Once this model OS have been built, we will customize the kernel code to add extra functionality that will involve user interaction.

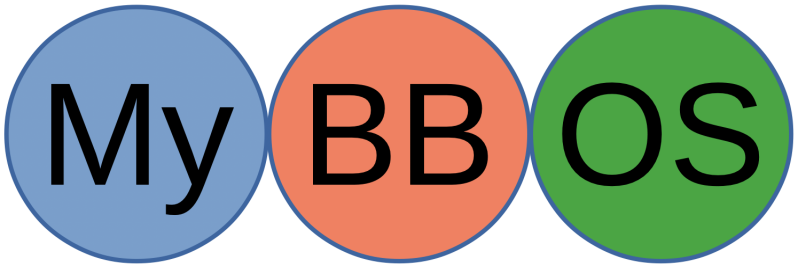

Figure 1: The flow for creation of threads from processes.

GitHub Starter Link

https://classroom.github.com/a/6ej2yejm

To use this link, please follow the steps below.

- Click on the link and accept the assignment.
- Once the importing task has completed, click on the created assignment link which will take you to your newly created GitHub repository for this lab.
- Clone this repository (bearing your name) and work on the lab locally.
- As you are working on your lab, you are to commit and push regularly. You can use the following commands to add a single file, you must be in the directory where the file is located (or add the path to the file in the command):
  - `git commit <nameOfFile> -m ‘‘Your notes about commit here’’`
  - `git push`

Alternatively, you can use the following commands to add multiple files from your repository:

- `git add -A`
- `git commit -m ‘‘Your notes about commit here’’`
- `git push`

Introduction

As a participation activity from last week, we spent time to program, compile and build a simple OS. The tiny OS was able to secure memory and to run in an infinite loop in an QEMU emulator, but not much else. In this work, we will again be working with a miniature OS called *BareBones* but this one is a little more complicated. The kernel of the current OS also secures memory as the previous OS project, however BareBones allows for messages to be written to the screen after booting and there is also ways to add to the kernel code to allows for customizations. In this lab, you are to follow the tutorial at `https://wiki.osdev.org/Bare_Bones` to build a working *BareBones* OS model. After this stage has been complete, you are to enhance and customize the code to implement some type of customized code to allow the user to interact with the OS. Your new code will be added to the file `kernel.c` and so it will be assumed that you spend time to understand how the kernel of the OS works.

Project Description

The OSDev.org website (`https://wiki.osdev.org/Main_Page`) is a rich resource for projects in building OSs and components. Here, you will find code and information concerning the implementation of a number of OSs, device drivers, booting systems, interrupt drivers and other important elements used to build an OS.

Locate this tutorial website and follow the *BareBones* OS documentation at (`https://wiki.osdev.org/Bare_Bones`) to implement its kernel and surrounding code. We will again be using a QEMU emulator to run the OS and so we will, hence, not be creating a the recommended bootable CDROM image to run the system. Please note, to ensure that all relevant libraries are present for the OS compiling and building, you will be compiling the kernel code in a Docker container. Find your `osdevi` Docker container in your `src/` directory. This same `Dockerfile` was also given out during *Participation 2* of last week and contains the correct compiler, libraries and emulation software to run your miniature OS.

After you have entered the code from the tutorial into each of the specific files, you will be ready to run the builder bash script file `builder.sh` to compile, link and build the OS. The builder file can also be found in your `src/` directory of your classroom repository, and can be executed by typing “`. build.sh`”. Please be sure that your source code is also available in this directory and, provided that your code is free of errors, your builder will initiate the emulator to run the OS. Note: Once you have are able to run the OS from the builder file (shown below), you can leave the emulator by typing `ESC+2` and then entering `q`.

The builder file is the following

```
#### START #####
```

```
nasm -felf32 boot_nasm.asm -o boot.o

i686-linux-gnu-gcc-8 -c kernel.c -o kernel.o -std=gnu99 -ffreestanding -O2 -Wall -Wextra

ld -m elf_i386 -T linker.ld -o kernel.bin boot.o kernel.o

qemu-system-i386 -kernel kernel.bin -curses
# Note: to exit from the qemu command, type ESC+2, then q and <Enter>

#### END #####
```

What TODO

Your tasks are outlined below.

1. Read through the tutorial at `https://wiki.osdev.org/Bare_Bones` to complete your code which is to be placed in your `src/` directory.

2. Locate your `Dockerfile` the `scr/` of your `classDocs` lessons directory. Build your container which will contain the compiler, linker and emulator software. The commands for this task are found in the comments at the end of the file. Note: building your container fir the first time may take some time.

3. Run the container using the file `src/run_osDevi.sh` which will also invoke the `sudo` privileges. To run this file, use the following command:

   . `run_osDevi.sh`.

4. Once you have a working, bug-free OS and have had a moment to play with it, you are to extend the code to add some new functionality that makes your OS *unique* in some way. This extension must allow the user to interact with the OS in some-way. For this user-interaction code, some ideas are the following:

   - A joke generator
   - Abstract poetry generator
   - A basic shell system
   - A username and password checking system
   - An implementation of threads to complete some user-defined task
   - A simple game
   - Some semaphores-driven mechanism

Since this is an extended lab, you have ample time to be creative in this deliverable of your OS.

Required Deliverables

This assignment invites you to submit an electronic version of the following deliverables.

1. File: `src/*`: The working code for your OS with its extended functionality
2. File: `writing/report.md`: A report document to show the output of your OS with your additional contribution that allows for some form of user interaction. Also in this report, you are to indicate how to run the code.

1 Honor Code

In adherence to the honor code, students should complete this assignment on an individual basis. While it is appropriate for students in this class to have high-level conversations about the assignment, it is necessary to distinguish carefully between the student who discusses the principles underlying a problem with others and the student who produces assignments that are identical to, or merely variations on, someone else's work. As such, deliverables that are nearly identical to the work of others will be taken as evidence of violating the Honor Code.