Phase 0 - Introduction to SPEDE!

Due Sep 18 by 11:59pm **Points** 100 **Available** until Sep 25 at 11:59pm

This assignment was locked Sep 25 at 11:59pm.

Introduction

Throughout our course, we will make use of SPEDE (System Programmers Educational Development Environment) which has been developed here at CSU Sacramento! This environment allows us to explore the foundations for systems programming and operating system concepts in an easy-to-use and easy-to-debug environment.

Before we can begin developing our operating system, we need to familiarize ourselves with the project environment. It is critical that each student completes this project phase independently so become familiar with the environment.

This project phase consists of four parts:

1. The SPEDE Virtual Machines
2. Customizing your development environment
3. Building/Flashing/Running/Debugging

When you have completed this project phase, you will have the foundations be ready to begin building your operating system.

Project Deliverables:

1. GitHub account registration / account submission
2. Console / teletype output to demonstrate the ability to utilize the SpedeHost environment for Part 3
3. Source code submitted to GitHub along with the console/teletype output

Part 1: The SPEDE Virtual Machines

The SPEDE consists of a few virtual machines used in this course:

1. SpedeHost - a full Ubuntu-Linux based operating system w/ the SPEDE host tools installed and ready to go

    1. We will use SpedeHost for serial terminals later in the semester
2. SpedeTarget - the target machine to run our operating system using the FLAMES envionment

The SpedeHost and SpedeTarget communication with each other using a serial interface. This allows us to download our operating system images, perform remote debugging, and more.

Note: the SpedeHost virtual machine should log you in automatically. If for any reason you need to log in (such as to perform an action as root, log in via SSH if you enable this, etc.) the following credentials are set up:

Username: spede

Password: spede

Minimum host system requirements:

1. Ability to run VirtualBox 6.1.x (Windows 7/10, OS X 10.8+, Linux on x86 CPU)
2. 10GB+ Hard Drive Space
3. 2GB+ RAM (1GB required for SpedeHost, 32MB required for SpedeTarget)

To download, install, and run the full lab environment, please follow the **SPEDE Virtual Machine Setup Guide.**

Launching the Virtual Machines

- Always start the SpedeHost virtual machine first
- Always start the SpedeTarget virtual machine second

Part 2: Customizing your virtual machine environment

While the SpedeHost virtual machine is generally ready to be used, there are a few things that you will need to do to prepare the environment for your own personal use.

You can launch a terminal from the desktop or from the applications menu

We will be using GitHub to submit each phase of the project. As such, you will need to configure git appropriately.

- Git configuration
    - git config --global user.name "Your Name"

- git config --global user.email "your.email@csus.edu"
- Set up your identity so any commits will be attributed to you

- Confirm your identification:
    - git config --list
- Generate your github SSH keys
    - See: **https://help.github.com/en/github/authenticating-to-github/adding-a-new-ssh-key-to-your-github-account** (https://help.github.com/en/github/authenticating-to-github/adding-a-new-ssh-key-to-your-github-account)
    - cd ~/.ssh
    - ssh-keygen -t rsa -b 4096 -f github.id_rsa -C "your.email@csus.edu"
    - cat ~/.ssh/github.id_rsa.pub
    - Add your public key to your GitHub authorized keys

If you are not yet familiar with git, you may wish to refer to the following references for basic usage:

- **https://education.github.com/git-cheat-sheet-education.pdf** (https://education.github.com/git-cheat-sheet-education.pdf)
- **https://ndpsoftware.com/git-cheatsheet.html** (https://ndpsoftware.com/git-cheatsheet.html)

The tools used to edit source code is highly personal. While vim and nano are pre-installed, you may feel free to install other tools that you are comfortable to use. You will be responsible for your own tools that you choose to use.

- Editors
    - emacs
    - gedit
    - Visual Studio Code
    - Atom
    - Sublime
    - etc.
    - vim and nano are pre-installed
    - You may install (at your preference) other editors; but you are on your own for any support!

Some additional customization that you may wish to consider:

- Suggestions:
    - Enable clipboard sharing (makes it easy for copy/paste between your host and the SpedeHost VM)

  - Enable Drag/Drop (makes it easy to copy files between your host and the SpedeHost VM)
  - Enable port forwarding for SSH access from your personal system to SpedeHost
- More!
  - Feel free to add additional tools and utilities as you desire, but keep in mind only the official environment may be supported
  - The SpedeHost VM hard drive is sized to 10GB, so be mindful as to what you install

Part 3: Building / Flashing / Running / Debugging

Summary of steps:

1. Clone the phase0 repository
2. Generate a Makefile
3. Build with debug symbols
4. Flash the operating system image
5. Running/debugging the operating system image

Cloning the repository

After you have launched and configured your virtual machine environment, you will need to accept the Phase0 assignment in GitHub Classroom and then clone the resulting repository:

https://classroom.github.com/a/BCwW3ku2 (https://classroom.github.com/a/BCwW3ku2)

Once the repository is cloned, continue with the remaining steps.

Generating a Makefile

Building an image is performed using the "make" tool. Therefore, you must have a Makefile that describes how to build the project.

Fortunately, this can be generated automatically for you by running "spede-mkmf"

```
$ spede-mkmf
An existing makefile was not found. Create one? (Y/n) y

Creating makefile 'Makefile' from '/gaia/home/project/spede2/Target-i386/i686/tools/etc/model.makefile'
```

```
The Makefile generated uses several variables that can be customized:
  * OS_NAME          - your operating system name
  * EXTRA_CFLAGS     - user specified compiler flags
  * EXTRA_LDFLAGS    - user specified linker flags
  * OPTIMIZE         - user specified compiler optimizations

See 'Makefile' for more information on these options.
```

A few notes:

- By default, your operating system name is "MyOS" but you can customize by editing the Makefile
- You can always create a new Makefile by running "spede-mkmf" again
- You can get a list of available make targets by running "make help"
- You can customize your Makefile, such as always enabling debug symbols via the EXTRA_CFLAGS variable

Some key build targets:

- make
  - Simply builds the project. DOES NOT include debug symbols
- make debug
  - Builds an image that includes debug symbols
- make clean
  - Removes all build artifacts
- make text
  - Generates the assembly source for your image
- make debug text
  - Generates the assembly source but also includes comments/C source for reference

Building with debug symbols

After generating the Makefile, build the OS image with debug symbols:

- make debug

This will generate a resulting operating system downloadable image (MyOS.dli).

Flashing the operating system image

Flashing the operating system image is the process of downloading the generated image file from the SpedeHost system to the SpedeTarget system. This can be performed using the 'flash' utility and using the built-in 'download' command

- flash MyOS.dli
- download

```
$ flash MyOS.dli
unable to open config file, using defaults....

>>>>>Welcome to the Flames Shell (FlaSh)  $Revision: 1.3 $ (SC/BJW)<<<<<<

Type ? for help
FLASH % download
File type is 'ELF'
Total blocks to download:  0x6b  (128 bytes each)

Load Successful ; Code loaded at 0x0x101000 (13760 bytes/sec)
```

Running/Debugging the Operating System Image

There are multiple ways to run the operating system once it is downloaded:

- Using the "FLames INTerface" (flint)
- Running through GDB

To run using flint:

```
FLASH % flint
FLAMES > g
```

To run using GDB:

```
FLASH % gdb
GDB386> continue
```

The flames interface can allow you to run the operating system image, inspect CPU registers and more. GDB can allow for a more robust form of debugging, including setting breakpoints, inspecting source code, and performing more user-friendly display/interaction.

As we are just beginning, let's try running the demo DLI using both methods. Note what happens when you press the 'b' key on the target console when running using each method.

1. What happens when pressing the 'b' key (on the SpedeTarget console) when running through flint
2. What happens when pressing the 'b' key (on the SpedeTarget console) when running through GDB

Please reference the SPEDE Lab Manual for more information on flint. For the most part, we will utilize GDB throughout the course when debugging, but we should ensure that our programs can run properly when not using a debugger.

Now, let's run through the debugger and demonstrate some basic GDB commands:

```
$ flash MyOS.dli
FLASH % download
FLASH % gdb
GDB386> continue
…
```

When the breakpoint is triggered, let's get a backtrace:

```
GDB386> bt
#0  0x001025c1 in breakpoint ()
#1  0x001011dd in main (argc=0x101080, argv=0x50408) at exercise.c:48
#2  0x0010113d in _start (memSize=0x2000000, parms=0x51136 "-g", loadedfiles=0x100000)
#3  0x00040ec1 in ?? ()
#4  0x00040320 in ?? ()
```

Let's look at the stack trace and move into frame 1 (where 'main') is running:

```
GDB386> frame 1
#1  0x001011dd in main (argc=0x101080, argv=0x50408) at exercise.c:48
48                            breakpoint();
```

Within the main function, let's inspect our local variables, such as:

```
GDB386> info locals
i = 0x196e6a
loop_count = 0x196e6a
col = 0x5
col_limit = 0x19
ch = 0x62
indicator = 0x2e
```

Now, modify different variables, such as loop_count, col_limit, or indicator and observe the results. Set breakpoints on different lines in GDB to explore the behavior. Step through the program line by line.

The full GDB user manual can be located here:

https://sourceware.org/gdb/current/onlinedocs/gdb/ (https://sourceware.org/gdb/current/onlinedocs/gdb/)

You may find the GDB Quick Reference to be useful:

http://athena.ecs.csus.edu/~cristg/resources/misc/gdb-quick-ref.pdf (http://athena.ecs.csus.edu/~cristg/resources/misc/gdb-quick-ref.pdf)