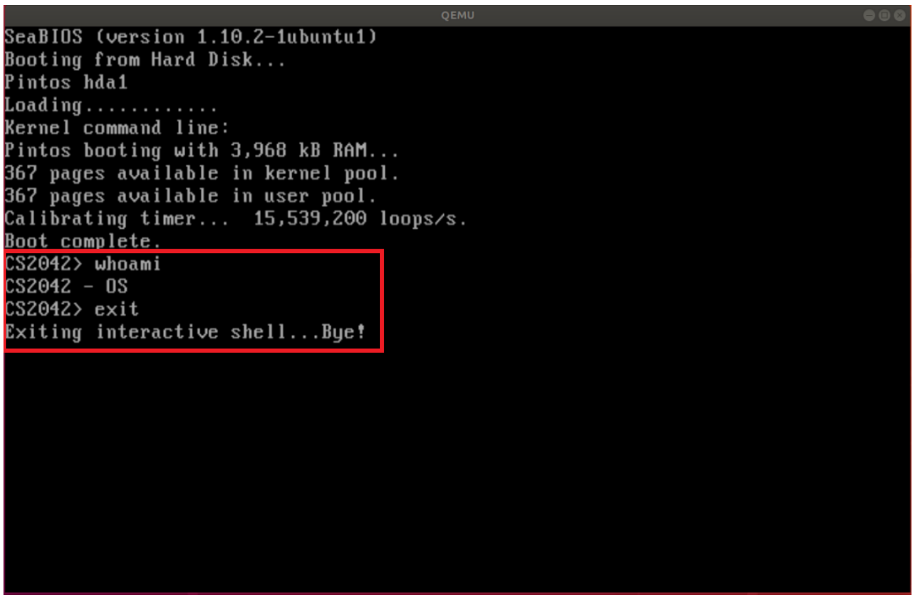
Interactive Shell for Pintos OS

CS2042 - Operating Systems

August 2021

1 Introduction

Almost every one of you has worked with a terminal/shell, whether it is an Ubuntu terminal, a Windows PowerShell, or some form of a Command Line Interface (CLI). In this exercise, we will introduce you to the behind-the-scenes of their functionality, well at least to a ridiculously simplified version of it. Before diving into the workings of a terminal, let's first understand the basics of its appearance and some terms that might be useful when implementing one. The following is how the Pintos OS terminal/shell would look like once you have completed this exercise, we will use that as an example.



```
SeaBIOS (version 1.10.2-1ubuntu1)
Booting from Hard Disk...
Pintos hda1
Loading...........
Kernel command line:
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer...  15,539,200 loops/s.
Boot complete.
CS2042> whoami
CS2042 - OS
CS2042> exit
Exiting interactive shell...Bye!
```

Figure 1: The Pintos OS shell

Here you can see our miniature shell in action inside the red color box. Each line starts with the string **CS2042>**, which is known as a prompt. The text that follows it is the command you (the user) gives to the OS. We have demonstrated to you two simple commands that our shell recognizes. Also, this is just a very crude implementation of a shell, feel free to modify it by adding a decorative banner at the top with a welcome message as you please.

2 Kernal Monitor

When Pintos finishes booting, it will check for any supplied command line arguments. In the future you will be using these arguments for testing purposes. If no arguments passed, the kernel will just finish running, which is good for testing but of no use for the user. Therefore, we will add a mini shell that will run when no command line arguments are passed.

One key fact that you should keep in mind is that this is a kernel-level shell. This is different from the normal programming that you have been doing so far! You will not have access to the standard C library since it works in the user mode. For example, when you call the `scanf` command, the function itself will call some system calls depending on the OS to get the job done. Hence, you will have to implement these functions by yourself when programming in the kernel mode. Luckily Pintos makes your life easier by providing a set of routines that mimic the standard C library. Although they are much simpler basic functions, you can use them as a foundation for your needs. If you are interested to learn more about this, you can get a more detailed description here.

3 Shell Commands

The objective of this activity is twofold. While you develop the shell, we want you to read through some of the important source files of Pintos OS to figure out how these functionalities are built into an OS. The following table contains the list of functions the shell should support alongside with a brief description of what they are supposed to do.

Command name	Description
whoami	Display your name alongside your index number
shutdown	Pintos OS will shutdown and exit the qemu emulator
time	Display the number of seconds passed since Unix epoch
ram	Display the amount of RAM available for the OS
thread	Display thread statistics
priority	Display the thread priority of the current thread
exit	Exit the interactive shell

4 Hints and Tips

Here are some tips that you can use as guidelines to get an idea on how to approach this exercise.

1. Try to read and understand `threads/init.c` file, which contains a placeholder to implement the shell in line 136.
2. You will find some functions available in `devices/input.c`, `lib/stdio.c` and `lib/kernel/console.c` useful.
3. Most of the functions that are needed for the shell commands are available in the `devices/` and `threads/` directories.
4. You can complete this exercise without including new files, therefore no need to edit the Makefile.
5. Put the knowledge you gained about buffers and string manipulation to the use.