Nachos Project Overview

(Adapted from Nachos documentation by Tom Anderson)

"I hear and I forget, I see and I remember. I do and I understand."
— Chinese proverb

1 Overview

The primary project for this course is to build an operating system from scratch. We believe that the best way for you to understand operating systems concepts is to build a real operating system and then to experiment with it. You will be provided with a very simple, but functional, operating system called Nachos. Over the course of the semester, your job will be to improve the functionality and performance of Nachos.

The project has five phases, corresponding to each of the major pieces of a modern operating system: thread management, multiprogramming, virtual memory, file systems, and networking. During this semester, we will complete the first three phases.

Some phases build on previous phases: for example, the file system uses thread management routines. As far as possible, we have structured the assignments so that you will be able to finish the project even if all of the pieces are not working, but you will get more out of the project if you use your own software. Part of the charm of building operating systems is that you get to "use what you build" – if you do a good job in designing and implementing the early phases of the project, that will simplify your task in building later phases.

Nachos runs on a machine emulation. It is important to realize that while we run Nachos on top of this emulation as a user program on UNIX, the code you write and most of the code we provide are exactly the same as if Nachos were running on bare hardware. We run Nachos as a user program for convenience: so that we can use gdb and so that multiple students can run Nachos at the same time on the same machine. These same reasons apply in industry – it is usually a good idea to test out system code in a simulated environment before running it on potentially flaky hardware.

In order to port Nachos to real hardware, we would have to replace our emulation code with a little bit of machine-dependent code and some physical machines. For example, in assignment 1, we provide routines to enable and disable interrupts; on real hardware, these functions are provided by the CPU.

Unless you work for a really smart company, when you develop operating system software you usually cannot change the hardware to make your life easier. Thus, you are not permitted to change any of our emulation code, although you are permitted to change any of the Nachos code that runs on top of the emulation.

Finally, a former student had the following suggestions for doing well on these programming assignments, and since we agree with all of them, we include them here:

Read the code that is handed out with the assignment first, until you pretty much understand it. *Start* with the ".h" files.

Don't code until you understand what you are doing. Design, design, design first. Only then split up what each of you will code.

Talk to as many other people as possible. CS is learned by talking to others, not by reading, or so it seems to me now.

2 Group Work

The first and second Nachos projects will be performed in groups of two. The third project will be performed individually. Group work was chosen for the first project so that you can work with another student to fully understand the Nachos code base. The coding for this project is fairly light. Group work was chosen on the second project because it is challenging and requires a significant amount of coding. Both members of a group should fully participate in both these first two projects in order to be in a position to successfully complete the third project. It will be very difficult to try to complete the third project, within the allotted time, alone unless you understand the implementation up to this point.

There is no restriction on collaboration within a group. Collaboration between groups follows the course collaboration policy, only "empty hands" discussions are allowed. See the course syllabus for more.

The same groups will be used for both projects one and two; groups cannot be reformed during the semester. When projects are performed in groups, there can be only one submission from the group. Further, all group members will receive the same project grade.

3 Dazzle

It is important to keep in mind the relationship between the functionality you have been asked to implement and the functionality of operating systems in general. Toward this end, five percent of each project score are "dazzle" points. These points are awarded for submissions that identify and implement functionality that is beyond the project requirements. Examples include an insightful and comprehensive test suite, implementation of new system calls (projects 2 and 3), or novel scheduling schemes. Any functionality that commonly exists in an operating system or that would be beneficial in an operating system is appropriate for dazzle points. **There is no collaboration between groups on the functionality that will be implemented for dazzle points. Each group must identify the functionality to be implemented on its own.**

4 Mechanics

The first step in doing the assignments is to make a copy of

`/classes/cs4411/code/nachos/nachos-projectA-2015.tgz`.

Untar the compressed file with the command **gtar xzf nachos-projectA-2015.tgz** and a directory `nachos/` will be created in the current directory. The `nachos/threads/` directory contains source code for the first assignment. **You may not** (and need not) modify the files in the **nachos/machine/** directory during completion of any of the projects. (This is the machine emulation code.) Note that this code runs **only** on Linux.

You should document your code if you want the reader to understand what you did (a requirement for getting a good grade!). Also, so that I can tell what changes you have made, you must surround every piece of code you modify with:

```
#ifdef CHANGED
   put your changes here
#endif
```

The following requirements apply to all submissions. Additional requirements may follow with each assignment.

- Use the `submit` command to submit each assignment. Each electronic submission should be created using the command `cd yourdir/nachos; gtar czvf projX.tgz *`, where X is the project number, 3 for the first Nachos assignment, 4 for the second, etc.
- Submit hardcopy of any file that you modify and the README. **Please highlight (with a marker) each `#ifdef CHANGED` and corresponding `#endif` on the hardcopy.** DO NOT submit hardcopy of unmodified files.
- Each submission should contain a README that gives the name of the group members, the number of slip days used (even if that number is zero), and the number of slip days remaining. The README should also enumerate what each group member did for the project. If you are asked to answer questions, the answers go in the README. The README should be in the `yourdir/nachos/` directory, not in a subdirectory (e.g., not `nachos/threads/`).
- Remember to put the slip days used for each project on the hardcopy.