CS350 : Operating Systems

General Assignment Information

1 Introduction

Assignments in CS350 are based on Nachos. Nachos is a workstation simulation, along with a simple operating system for the simulated workstation. The assignments require you to enhance the Nachos operating system. For each assignment, you will be given a set of general requirements describing the enhancements that must be made. You are to design, implement, test and document changes to Nachos that will satisfy the requirements. You will also be required to demonstrate that your system behaves as required by providing **readable** and comprehensive testing documentation.

2 Using Nachos

Nachos is available in the CSCF Unix environment. To use Nachos, you must first install it in your account. There is a shell script, called `install_nachos`, that will do this for you. Please read the Nachos installation instructions on the course web page and follow them carefully.

You may work on your implementation on machines outside of the CSCF environment. In particular, see the course web page for information about running Nachos on Linux. However, **to receive credit for your implementation work, your code must compile and execute correctly in the CSCF environment. It is your responsibility to ensure that it does so.**

The course web page also contains important information about using Nachos, about working in groups and sharing files in the CSCF environment, and about Nachos itself: how the machine works, how the operating system is organized, and what it is capable of doing. Please read it.

3 Project Groups

You may work on these assignments alone, or in groups of up to three students. You are responsible for completing the assignments whether you have partners or not. The requirements are the same in either case. If you choose to work with partners, they need not be in the same section as you.

If you want a partner and do not have one, you may wish to try posting a "partner wanted" message on the course newsgroup.

If you work in a group, you must apply to us for a Unix group identifier. Apart from being just an administrative requirement, this will also help members of your group to share files. See the course web page (under Working in Groups) for information about various ways of sharing files in a Unix environment.

Choosing your group and obtaining a Unix group name is Assignment 0. Follow the Assignment 0 instructions on the course web page.

4 What to Submit

For each assignment you are expected to submit the following items:

Cover sheet/marking guide

The cover sheet/marking guide for each assignment can be downloaded and printed from the course web page. Each assignment has its own cover sheet/marking guide. **A stiff penalty will be inflicted on submissions that do not include this cover sheet.**

Design document (3 pages maximum)

Your design document should include:

- A description of how you enhanced the Nachos operating system to satisfy the assignment requirements. For example, you might discuss significant new classes you added, explaining what they do, how they do it, and how they are used to implement the assignment requirements.
- A brief description of what works and what does not, clearly and explicitly indicating any parts of the design that have not been implemented.
- A discussion of strengths, weaknesses, and limitations of the design, and justification for the design choices you made.

It should **NOT** include:

- a restatement of the assignment or any portion of the assignment.
- sections of code
- code showing class definitions, or lists of function or method prototypes.

Your design document should be entirely self-contained. A good way to think about how to write your design document is to aim that readers should get an excellent understanding of your system's design, without referring to your code submission.

Testing document (3 pages maximum)

Your testing document should includes:

- brief overview of your testing strategy
- testing issues (what it is important to test)
- testing plan (how you arranged to address each of the testing issues)
- brief description of how to run the tests and a description of the output that is expected.

Your testing document, like your design document, should be self-contained.

It is not acceptable to "trade" testing document pages for design document pages, or vice versa. For example, it is not OK to submit a four page design document if your testing document is only two pages long. *Each* document has a three page limit.

Code

Your Nachos code and test programs. Do a `make distclean` in your Nachos build directory before submitting your code. There is no point to including all of those `.o` files and executables in your submission, since we are going to rebuild your system anyways.

Your cover sheet/marking guide, design document, and testing document must be submitted in hard copy form in the CS350 assignment boxes on the 4th floor of MC by noon on the due date. Your Nachos code and test programs must be submitted electronically using the `submit` command. Do not submit hard copies of your code.

4.1 Design Document

There is a hard limit of three pages for this document. Use a readable font, at least 10 point. Longer documents submitted to us will simply be truncated after three pages.

Your design document should provide an overview of the changes you have made to Nachos to support the assignment requirements. Write your document for an audience that already understands operating systems in general, Nachos in particular, and the assignment requirements. Assume your readers will be asking *how?* and *why?*, and provide answers to these types of questions. Your document should explain how each of the assignment requirements were addressed in your system.

An important aspect of the design document is justifying your design decisions. Sometimes one is forced to make decisions to solve certain problems and other times one makes a design decision in anticipation of future extensions and demands on your code. We want to know how such things affected your design.

If your design does *not* address some of the requirements, those that are not supported should be noted explicitly. Finally, if your system implements features other than the required ones, your document should describe the extra features, and should explain how they were implemented.

Your design document should not include program code. If you wish, your document may refer to specific parts of the code for additional details. However, your document should be self-contained. The markers should be able to determine whether your design addresses all of the assignment requirements without having your code in front of them.

You may find it helpful to have at least an outline of your design document prepared for your group before beginning heavy coding. This way your group members have a better idea of what their tasks are, and your group can coordinate its efforts more effectively. Also, if you write your design first in English, you may find it easier to implement it in C++, rather than the other way around.

NOTE: your group is responsible for ensuring that the design document matches the implementation and visa versa. Any description of features or designs that are implied to be implemented but are not actually implemented will be treated as a case of academic dishonesty It is acceptable to explain the design for unimplemented features provided that it is clearly and explicitly stated which features were not implemented. .

4.2 Testing Document

You are required to provide a set of user test programs to demonstrate the functionality of your Nachos system. In addition, you must produce a testing document. This is a document that tells the markers what aspects of the system you have designed tests for.

As is the case for the design document, you may design tests and describe them in your testing document even if they are not implemented. However, your document must clearly identify which of the described tests, if any, are not implemented.

There is a hard limit of three pages for this document. Longer documents will be truncated at three pages.

Your testing document should clearly state which aspects of your system you are trying to test, and it should describe how each of these aspects is tested. To describe how some aspect of the design is tested, you will need to identify the test program (or programs) that accomplish the test, what that program actually does, how that program should be run to accomplish the test, and the output that should be expected. Ideally, the output itself will be self-explanatory, but if explanation is needed, it should be included.

We do not expect 100% test coverage of your implementation but we do expect you to do a reasonable job of designing tests that will convince us that your implementation is correct. Your testing plan should cover the main features of your design. You should also include some testing of error and boundary conditions.

4.3 Code

You are required to submit a complete and self-contained copy of Nachos, modified as described in your design document to meet the assignment requirements. This should include both the Nachos code itself, as well as the user (test) programs you have written. We will build your submitted code, and then use your system to run your test programs.

Your code should be submitted on-line, using the `submit` command. For more information on this command, type `man submit` . When you run this command, you should be in the top level directory of the copy of Nachos that you wish to submit. The top level directory is the one that has `code` and `c++example` and `coff2noff` as subdirectories. Do a `make distclean` before submitting your code to remove binaries and similar compiler waste products (we will recompile your code on our own).

5 Marking

For each assignment your mark will be based on your design, your testing, and your implementation. as outlined in the marking guide. The design portion of your mark will be determined by your design document. The testing portion of your mark will be determined by your testing document and test programs. Test marks are awarded if you have understood what needs to be tested, and if you have done a good job designing

and documenting effective test programs. The implementation portion of your mark will be based on how well your system (including the test programs) operates. If you have a good design on paper but the implementation was sloppy and/or has serious flaws you should receive a good grade for the design but a lower grade for the implementation. The implementation portion of your mark will be determined by the execution of your test programs, an review of the code, and by the execution of additional test programs by the markers. There will be *no* implementation marks for code that does not run or that cannot be tested. This means that you should implement and test one part of the system at a time, rather than doing all the implementation first and leaving the testing until the end.

Your design and testing documents are expected to be well-organized and clearly written. Your code is expected to be well-structured, commented and readable.

5.1 Academic Dishonesty (a.k.a. cheating)

You are encouraged to discuss the course assignments with people outside of your group and to use the course newsgroup for such discussions. **Nevertheless, each group is expected to do its own detailed design, to prepare its own documentation and to do its own implementation and testing.** For example, it is okay to discuss why Nachos (as given to you) behaves in a certain way, or why you cannot get it to compile, or how to use its debug mode, or what a semaphore is, or the differences between two paging algorithms, or the problems that arise when a multi-threaded process is terminated. It is *not* okay to share the Nachos code that implements process termination or the design documentation that describes it. It is the responsibility of each group to ensure that its on-line code and documentation are protected from general access.

A good guideline is to leave pencils and paper (and their electronic equivalents) behind if you discuss the assignments with other groups.

A related matter is the acquisition of modified Nachos code from students who have taken this course before, in a previous term. The are several problems with using such material to aid you in your own work:

- This material is unauthorized and its use falls under the category of plagiarism.
- More often than not, every term, assignment requirements are subtly changed and assumptions are revised. We make updates to the initial Nachos code as well. These changes make the previous assignment submissions a potential source of misguidance that may cost you dearly in correctness marks, should you choose to use them.
- We archive **all** student submissions from previous terms and include them when running our cheat detection software. So, the chances of our discovering your use of previous submissions are quite high.

The standard penalty for cheating is a grade of **-100%** on the assignment. Any such incidents will also be reported to the Associate Dean (Undergraduate Studies) of the student's faculty. This may lead to additional punishment.