Teams:

The project is to be done in teams of three (by this point, you have already been assigned to your team). Once a team is formed, it may not be dissolved except for the most bizarre and unusual reasons. If you have problems within your team, work them out internally. Team members will generally all receive the same grade for each assignment, although I will give each member of a team the opportunity to judge the contributions of other members of that team to the assignment (see last section of this document).

Slip Days:

You will have flexible slip dates for the assignments. Each team is given an automatic extension of 5 calendar days (not class days), which can be used on any Nachos programming assignments, in any integer increment, as long as the total amount of lateness does not add up to more than 5 days. You must use your slip days in integer increments — so, for example, if you submit your assignment 12 hours late, that counts for one slip day, not one-half slip day. (Note that any school holidays, e.g., spring break, also count against your slip days.) This should let you schedule due dates around the due dates for other courses. When you hand in an assignment, you must identify at the top of the assignment, (i) how late this assignment is, and (ii) how much of the total slip time you have left. After you have used up your slip time, any assignment handed in late will be marked off 20% per day. There will be no extensions granted.

Revision Control and Backup:

You are completely responsible for revision control and backups. For revision control within your team, you may want to take advantage of the capability of groups in Linux (once teams are determined, I will have a group set up for you) along with revision control software such as CVS. (CVS – and its ancestor RCS – appears to be the only thing installed on our systems currently).

Coding Specifications:

The nachos1/ directory contains source code for Nachos for the first assignment in the directory threads/, plus code for the machine emulation in the directory machine/. For the first assignment, you will need to make changes to files in threads/ (additional directories will be provided for later assignments), but you should not modify the files in the machine/ directory. Also, so I can tell what changes you have made, surround every piece of code you modify with:

```c
#define CHANGED
// put your changes here
#endif
```

Note that once you have done this, you will need to have the line

```c
#define CHANGED
```

in each file (or a common header file) in order for your changed code to be compiled in.
Write-Ups:

In addition to your code, I expect the team to submit a written outline/description of what you did for the assignment, explaining and motivating your design choices. Keep the description short and to the point, but also make sure that it is complete. Of course, you must document your code if you want the reader to understand what you did (a requirement for getting a good grade!). Some assignments have parts that ask you to explain something about your design; this explanation must also go in your outline/description.

I expect you to run test cases to demonstrate that your code works (or that it doesn’t!). The results of these test cases must appear in a test-case description that accompanies the outline/description above, and includes an explanation of what each test case is supposed to show. **I will not tell you what test cases to run — so do not ask.** It is up to you to convince me that your code works. You should pick the minimal set of test cases that do so.

Submit your write-up as a PDF, included in your tarball.

The electronic document must be placed in your top-level `nachos1/` directory.

Submission:

To submit an assignment, create a tarball of your `nachos1/` directory and all contained files using

```
tar -czvf nachos1-<teamname>-submit.tgz nachos1/
```

where your team name replaces `<teamname>`. One member of the team must then submit the tarball to my netfiles inbox.

Grading:

The grading for the project will be as follows: 35% design, 50% implementation, and 15% “professional polish” (see below). I have structured the grading in this way to encourage you to think through your solution before you start coding. If all you do is work out a detailed design for what you would do to address the assignment (and if the design would work!), but you write no code, you will get 35% credit. The implementation portion of the grade considers whether you implemented your design correctly, ran reasonable test cases, and provided documentation that the reader can understand. Finally, to encourage you to come up with good designs (rather than ones that only work), I am assigning 15% of the grade for each assignment to “professional polish points”. The base grade is 85% if your design and implementation work perfectly; you may get additional points if your solution is elegant, simple, and easy to understand, or if it goes well beyond the requirements of the assignment. If you think you deserve “professional polish points”, you must convince me of this in your write-up.

Individual Assessment Write-up:

Each member of the team must also submit an individual write-up. In this write-up, list in detail those aspects of the project for which you are completely responsible; also list those aspects for which you assisted in development.

Also, I will give each team member an opportunity to evaluate the amount of work that other members of the team gave to the project. Specifically, given 20 points total, each team member should divide the points among all team members (including yourself). For instance, if you feel that both members of your
group did the same amount of work, give 10 points to each member. If you feel that you did three quarters of the work and your partner did only one quarter, give yourself 15 points and your partner 5. You must provide some brief explanation about why you assigned the points in the manner that you did—briefly explain the division of work.

Do this in private—I will keep in confidence (to the extent possible) each individual’s assigned participation points, but will use the points to determine each member’s ultimate grade. I expect (hope) that each team member will obtain the same grade on each project, and I expect everyone to work hard to be sure this is so. This mechanism allows me a way to address any unusual situations. Send the assessment to me via email as an electronic attachment (preferably PDF).