CS162, Spring 2004 **Discussion #1** **Amir Kamil**
UC Berkeley **1/22/04**

Topics: Administrivia, Course Overview, Introduction

1 Administrivia

1.1 About me:

- Amir Kamil
- 4th year EECS major
- Email: `kamil@eecs.berkeley.edu`
- Webpage: `http://www.cs.berkeley.edu/~kamil`
- Be warned that I have a twin brother who is a CS major (and is taking this class), so if you get an unexpected reaction when you ask one of us for help, it's probably him.

1.2 Class logistics

- **Section assignments are temporary.** You will be assigned to a new section based on your project group.
- Project groups: 4-5 people in each group, **all of who must be able to attend the same section**.
- Sign up for project groups immediately on the web page.
  1. Form a project group of 4-5 students, who can all attend the same section.
  2. **You cannot use this form until every member of your group has received an account form, logged in to the account, and registered by running `register` (this should be run automatically the first time you log in).**
  3. Run `re-register` if you did not put your name in properly when you originally logged into `cs162-*` account.
  4. Each project group must fill out one copy of this form.
  5. We will then assign project groups to sections.
  6. If you cannot find project group, email me with your available times, or post to the newsgroup.

1.2.1 Nachos

- Used for our projects, by ex-Berkeley prof
- Great system, easy to use, makes OS development experimentation much easier than hacking a real OS
- Implemented in Java
- Runs on UNIX, Windows, etc - very portable
- Nachos roadmap online
- Please, do not print the Nachos code, use your reader.

1.3 CVS

CVS is a project management system that you will be using for this class. Each group will be given a repository, which both allows you the store your files in a central location, and backup previous versions of your code. There is a CVS help session webcast on the class webpage, and we will go over the basics next week in section.

1.4 Contacting staff

Any class or project related questions should be posted to the newsgroup, `ucb.class.cs162`. Please check to see if a question has been answered before posting. Google search (`groups.google.com`), though delayed, should be useful for this. For personal question, you can email the entire staff at `cs162@cory.eecs`, or me at `kamil@eecs`. Note that the latency for answering a question is least for the newsgroup, and most for emailing me directly.

2 Course Overview

The short list of topics we will cover in this course is:

- operating system structures
- concurrency, threads, synchronization, synchronization primitives, deadlock
- scheduling
- address spaces, caching, TLBs, paging
- file systems, disk management
- networks, distributed systems, network protocols

A secondary goal of this class is to teach you software engineering and group management through the projects.

3 Introduction

3.1 History of operating systems

We begin by briefly looking at operating system history. Besides being interesting, studying OS history not only will help us avoid repeating the mistakes of the past (e.g. Windows 3.1), but also illustrates the different design decisions made based on different requirements and goals. OS history can be divided into 5 major eras.

1. Late 1950's
   In this period, humans are cheap relative to machines. The job queue is a stack of cards. The OS would load the job print the job, and make a dump of errors. Only one job at a time could run (no spooling). Languages were COBOL, FORTRAN, and ALGOL.

2. Early 1960's
   Mostly batch processing, where submitted jobs wait in a queue to run. Basic multiprogramming. Many jobs can run at the same time. Overlapping of I/O and computation. Use of *spooling* so the system is not tied down to I/O devices. Protection provided because of sharing. MULTICS (1963-1969) is a fully-featured OS that took 6 years to develop. Hardware is still expensive.

3. 1970's
   Hardware becomes cheap. People more expensive. Interactive time sharing systems developed. Fancy filing systems. Minicomputer (PDP, PDP 11). UNIX OS.

4. Early 1980's
   Personal Computing. Networking, distributed computing.

5. Mid 1990's
   Mobile Systems (laptops, PDAs).

3.2 General operating system functions

The definition of *operating system* varies depending on who you ask. Microsoft, for example, will tell you that an internet browser is a crucial component of an OS, will the US Department of Justice will tell you otherwise. It is easier to define what an operating system *does*, rather than what it *is*. There are two main operating system functions:

1. The operating system is a *coordinator of resources*.
   It manages the resources of a system, allowing multiple programs to share a resource. In addition, it provides protection between programs so that one doesn't destroy another.

2. The operating system is a *facilitator* or *extended machine*.
   It provides standard services and shared libraries, such as input/output routines, that all programs and users can use without having to implement them from scratch.

An internet browser could classify as a standard service, so it could make sense for the operating system to provide one.

3.3 Operating system design goals

Operating system development is guided by some design goals that influence what is implemented and what algorithms are used. Some general design goals are:

1. Convenience
   This includes convenience for the applications programmer and for the user.

2. Efficiency
   In some systems, such as supercomputers, hardware resources are very expensive. In such a system, it is desirable to make the most efficient use possible of the resources, keeping them in use as much as possible.

3. Fairness
   The operating system should be fair to all programs and users in allocating and scheduling resources. Their are different definitions of fairness, however, which we will discuss when we look at CPU scheduling.

4. Priorities/Deadlines
   Some programs should be preferred by the operating system over others. For example, if you are watching a movie while downloading something off the internet, you'd prefer that your download takes a little longer over your movie skipping frames.