CS-450 & CS-550: (Operating Systems)
Fall 2005 Semester

Term Project Topics and Coverage

TOPICS: Your semester project can be either of two types. First of all, you can choose either an actual commercial or an experimental operating system, perform library/internet research on it, and then report on it. Several examples of acceptable operating systems are listed below. You can choose one of those listed, or any other operating system not listed, except that: (i) Neither Microsoft Windows in any of its variations nor ***UNIX/Linux*** or any variant thereof may be selected; and (ii) at most **one** group from each section may report on any single operating system.

Secondly, you may **propose** any other topic of your choice related to Operating Systems. Send an E-mail to me with the subject header ***OS-450*-n *Semester Project Topic* or *OS-550*-n *Semester Project Topic,*** and describe your topic in one paragraph or less. I will get back to you as soon as practicable with a message of either acceptance or rejection.

GROUP COMPOSITION: Each group will be composed of students from one section only. For undergraduate students (CS-450), each group will consist of **at least four** and **at most five** students. Put together your own groups, but I reserve the right to impose a change in group composition if necessary, so that I can assure that each student has a group while at the same time group size limitations are maintained. For graduate students (CS-550), each group will consist of **exactly one** student.

Nailing Down Your Topic Selection and Group Composition: There will be a separate sign-up sheet posted outside my door for each section. Fill out in ink on the topmost blank line **on the appropriate sign-up sheet for your section:**

(i) Your topic selection (OS1, OS2, . . .), and also
(ii) (ii) list the last names of **all** members of your group.

Check **before** you fill in your topic to make sure that no one else has already selected it.

Should you decide to change your topic, then:
(i) **cross out** (do **not** erase)your first entry;
(ii) go to the topmost blank line.; and
(iii) write in a new entry corresponding to your new topic.

OSs:

1. Amiga Operating System
2. Apple's MacIntosh O/S (latest version)
3. IBM OS/2 Warp
4. IBM's OS/360, OS/370 or OS/390
5. IBM's VM
6. DEC's RSX (for the PDP-11)
7. DEC's RSTS (for the PDP-11)
8. DEC/Compaq/Hewlett-Packard's VMS
9. MS-DOS/PC-DOS
10. Mach (originally from Carnegie-Mellon University)
11. VxWorks
12. PalmOS
13. Xoberon (a real-time OS)
14. Quadros-rtxx (a real-time OS)
15. rt-Linux (a real-time OS)
16. lynx (a real-time OS)
17. MIT's Exokernel (http://www.pdos.lcs.mit.edu/exo.html):
    "An exokernel eliminates the notion that an operating system should provide abstractions on which applications are built. Instead, it concentrates solely on securely multiplexing the raw hardware: from basic hardware primitives, application-level libraries and servers can directly implement traditional operating system abstractions, specialized for appropriateness and speed." We would focus on how the exokernel offers advantages (and certain disadvantages) over the traditional microkernel and monolithic architectures. Operating systems used to

compare could be Mach (microkernel) and Linux or Windows (monolithic). There is also a student-made smaller OS called Adrenaline which is based on Exokernel.

18. SCOUT Operating System (http://www.cs.arizona.edu/scout/)
"The ubiquity of computer networks is fueling an explosion in small, special-purpose, and often mobile computing devices. These devices, which are sometimes called network appliances, include network-attached disks, cameras, and displays; set-top boxes and web browsers; hand-held and portable devices; application gateways; and special-purpose servers (e.g., web and file servers). Network appliances have several unique characteristics that suggest a re-thinking of operating system design." The characteristics listed and that would be focused on are
--Communication-oriented
--Specialized/Diverse Functionality
--Predictable Performance with Scarce Resources

19. TUNES (http://tunes.org/Tunes-FAQ-4.html#ss4.1)
"To sum up the main features in technical terms, TUNES is a project to replace existing Operating Systems, Languages, and User Interfaces by a completely rethought Computing System, based on a fully reflective architecture...Prominent features built around this reflective architecture will include unification of system abstractions, security based on formal proofs from explicit negotiated axioms as controlled by capabilities, higher-order functions, self-extensible syntax, fine-grained composition, distributed networking, orthogonally persistent storage, fault-tolerant computation, version-aware identification, decentralized (no-kernel) communication, dynamic code (re)generation, high-level models of encapsulation, hardware-independent exchange of code, migratable actors, yet (eventually) a highly-performant set of dynamic compilation tools." This OS appears to be highly experimental; there is not even a compiling version yet; however it is revolutionary in that it breaks free of traditional and potentially highly inflexible OS architectures.

20. SPIN (http://www.cs.washington.edu/research/projects/spin/www/)
"SPIN is an operating system that blurs the distinction between kernels and applications. Applications traditionally live in user-level address spaces, separated from kernel resources and services by an expensive protection boundary. With SPIN, applications can specialize the kernel by dynamically linking new code into the running system. Kernel extensions can add new kernel services, replace default policies, or simply migrate application functionality into the kernel address space. Sensitive kernel interfaces are secured via a restricted linker and the type-safe properties of the Modula-3 programming language. The result is a flexible operating system that helps applications run fast but doesn't crash." Similar to MIT's exokernel in many features, primary the notion of extensibility of the kernel (similar to 'exokernel' concept).

To be addressed for each operating system, as time and personal resources permit (listed in decreasing order of importance):

(1) Overview: In what environment is your O/S designed to work? Did this O/S advance the state-of-the-art? Is it or was it commercially and economically successful? technically successful? What was done right with it, and what was done wrong?

(2) Processor modes and privileged instructions, uniprocessor or multiprocessor, SMP or master/slave?

(3) Allowable process states (map the terms used by the OS to the generic states that we covered in class)

(4) Memory management: memory model, and implementation details

(5) Data structures used for process management, for memory management, and for file management

(6) CPU Scheduling

(7) Singly- or multiply-threaded per process? Kinds of threads, if applicable

(8) How is deadlock dealt with? Deadlock Prevention, Deadlock Avoidance, Deadlock Detection-and-Resolution, more than one of the above, or none of the above?

(9) File management

(10) Facilities provided for mutual exclusion and for synchronization