Administrivia

- **Last project due today**
- **Final Exam**
  - Wednesday December 12, 12:15-3:15pm
  - Right here in Gates B01
  - Open book, open notes, just like midterm
  - Covers material from all 19 lectures
- **I have special office next Monday, 2:45-3:45pm**
  - I also plan to be around most of the afternoon that day, so stop by if you have questions before exam
- **Televised question session tomorrow 4:15pm-5:05**
  - Please come and bring any questions you might have on lecture material

Themes in OS research

- **Performance**
- **Functionality**
- **System management**
- **Extensibility**
- **Power consumption**
- **Security [guest lecture]**

Performance

- **Performance improvements always welcome**
  - 10% is nice, but often not super interesting
  - 10x can actually enable new functionality
- **Through early 90s, a major focus of OS research**
  - Makes benefits nicely quantifiable
  - Let to lots of incremental work
- **But some performance work very interesting:**
- **Synthesis [Massalin]**
  - OS made extensive use of dynamic code generation for speed
- **Making logically synchronous disk accesses asynchronous [Nightingale]**
  - Make *fsync* instantaneous until effects *externalized*
  - Assume NFS cache consistent, roll back execution if wrong

Functionality

- **Lots of work to make distributed systems transparent**
  - Network operating systems (Sprite, Amoeba)
  - Distributed shared memory (tons of papers)
  - Much of this work had little impact
- **Plan9 [Pike] – make all functionality available through file namespace**
  - Developed by the inventors of Unix, who gave up on Unix
  - Invented now popular abstractions such as `/proc`
  - Mount table is no longer global, but per process group
  - Lots of really cool benefits to unifying abstractions in FS:
    - Bypass firewall by remotely-mounting `/net`
    - Window system just virtualizes console device

System management

- **Already talked about virtual machines**
- **Plan9's features greatly simplified management**
  - E.g., backups available through file system under `/dump`
- **Storage management a huge deal**
  - Large, virtual disks (e.g., Petal [Lee])
  - Serverless network file systems [Dahlin]
  - Using desktop machines for storage (Farsite [Bolosky])
  - Peer-to-peer network file systems (e.g., Ivy [Morris])
  - Peer-to-peer backup (e.g., Pastiche [Cox])
- **Application abstraction (Singularity [Hunt])**
  - Avoid problems associated with installing software
  - Packages must describe themselves declaratively
  - Applications are a security principal that can go in ACLs

Extensibility: Microkernels

- **Very popular in 1980s**
- **Idea: Provide traditional OS abstractions in servers**
  - E.g., Virtual memory server, file system server
  - Could Make for better fault isolation
  - Also makes it easier to develop new OS functionality (debugging servers potentially easier than debugging kernel)
- **Kernel interface is very small**
  - E.g., just provide simple IPC abstractions
  - Note: "micro" means small interface, not small code
- **Most well-known example: Mach 3**
  - Big (0.5 M lines of code in 80s) and bloated
  - Performance problems in part from context switches
  - Influential system but gave microkernels a bad name
- **More successful microkernels: VxWorks, L4, MINIX3**

Extensibility: Other architectures

- **Big focus in mid-90s:**
  - Inspired by applications that fight with the OS
  - E.g., database that has to bypass the OS buffer cache
  - Also CPUs still slow enough that hard to saturate net
- **One approach: Spin [Bershad]**
  - Download extensions into kernel using safe language
  - Safely run user code in kernel, saving context switch
- **Another approach: Exokernel [Engler/Kaashoek]**
  - "Exterminate all operating system abstractions"
  - Idea: Kernel should provide protection, not abstraction
  - Implement abstractions in user-level library
  - Replacing, e.g., socket implementation like replacing malloc

Power management

- **Recent hot topic for sensor networks**
  - Battery-powered devices
  - Deployed in environments where hard to change battery
  - OS techniques can extend battery lifetime from days to months
- **Also becoming an issue for server farms**
  - Cost of power + cooling comparable to cost of hardware
- **Some techniques**
  - CPU voltage scaling (requires cutting frequency—when to do?)
  - Careful use of wireless networks
  - Probably will see more with disks in near future

Final thoughts

- **You are all now operating systems experts**
- **Use this knowledge to build better applications**
  - Sometimes need to coax right behavior out of kernel
  - Should be much easier now that you know what's going on
- **Syscall interface can be an *innovation* barrier**
  - Much harder to change kernel than user code
  - Other examples include standardized net. protocols, servers
  - Get these wrong and many people will suffer
- **Some of you will go on to design interfaces that many people are later subjected to**
  - Strive to achieve both simplicity and flexibility for users

How to learn more about OSes

- **Take CS240 – Advanced Topics in Operating Systems**
  - Class will bring you up to speed on OS research
  - Read & discuss 18 [Mazières] or 25 [Engler] research papers
  - By the end, should be ready to do OS research
- **Get involved in research!**
- **Lot's of interesting OS work at Stanford**
  - Rosenblum – launched the virtual machine resurgence
  - Lam – collective system
  - Levis – seminal work on sensor nets & power management
  - Engler – tools to find OS bugs automatically
  - Boneh/Mitchell – lots of practical OS security work
  - Mazières – file system and security research
  - Today will hear about new HiStar OS [Zeldovich]