Outline

Objective:
– Summary of course

Parallel Processes

• Interrupt-driven code dictated by I/O device requirements
• VERY hard to debug
• Cooperating sequential processes a good alternate model
  – Need synchronization
    • Critical sections most important: They allow new ATOMIC OPERATIONS to be defined
  – Deadlock
    • Avoid via ordered resource requests
    • Banker’s Algorithm considered too conservative for use

Scheduling

• FIFO, SJF used before operating systems: Factory scheduling
• Interactive systems pose problems
• “Ideal” schedules need to know future
  – Possible if “same” jobs reoccur
  – Periodic time-deadline tasks
• Scheduling multiprocessors NP hard
  – So is scheduling super-scalar processors
  – Approximate
Memory

- Virtual memory allows more programs to run in parallel
  - May let them interact with slow human
- VM management algorithms
- Best solution: HUGE DRAMs

Files

- Abstraction simple to use
- Allows many devices to be used through same interface
  - Modern OS’s allow device drivers to be added later. UNIX requires them to be linked into OS kernel
- Files allow disk to be used efficiently

Protection

- When multiple programs run together, nice to be able to keep one from “damaging” another
- Relocation/limit registers
- Paged VM
- Careful separation of user and kernel
- Timer interrupts keep one program from hogging whole CPU forever
Security

- New dimension to protection – across machines, even across Internet
- Based on Cryptography
- Weak link: Authentication
- Also adds overhead computation, but this isn’t serious
- With proper policies, allows controlling leakage of information, restriction of modification rights