OS Components

Son Le
Memory management

• Present on most conventional OS
• Each process has
  – its own address space
  – (a feeling of) contiguous memory
• Pros
  – Security: a process’ memory is separated from the others’
  – Facilitating application programming
• Cons
  – Requiring dedicated hardware to be efficient → page replacement strategy?
  – External storage is also needed
System calls

• A mechanism for a program to request service from an OS’ kernel
• Separation of user level from kernel level: a desired feature
• Implemented using interrupt or dedicated instructions
• Some operating systems wrap system calls inside API (Windows) – desirable if there’re many applications
TinyOS

- No kernel, more like a programming library
- Single process
- Multithreading since TinyOS 2.1
- No virtual memory
- No dynamic memory allocation
- Powerful supporting toolchain
TinyOS’ threads

• Fully preemptive, round robin, 5ms time sliced
• Fixed number of threads
• One kernel thread & multiple application threads
• Kernel thread has higher priority over application threads
• Application threads run with the same priority
TinyOS’ threads

1. Boot up
2. Initialize
3. Boot up code
4. Pick a task
5. Run task
TinyOS’ threads

Timer Interrupt

Schedule thread
If any task is waiting

Pick a task
Run task

Otherwise
Post a task

Issue blocking syscall

Thread Scheduler
Kernel Thread
Task Scheduler
Application Thread
Syscall API
Task
TinyOS’ threads

- Thread Scheduler
- Kernel Thread
- Task Scheduler
- Application Thread
- Syscall API
- Task

Interruption

Process data

Post a task

Pick a task

Run task
Mantis OS’ threads

- Fully preemptive, round robin, time sliced
- Fixed number of threads
- Single process
- Unix-like thread scheduler
Mantis OS’ threads

- Thread Scheduler
- Ready List
- Semaphore List
- Application Thread
- Device Driver

Timer Interrupt

Run thread

Post a semaphore

Hardware Interrupt

Process data