Where Have We Been?

- Kernel Services
  - Task Dispatching
  - Task Scheduling
  - Intertask Communication (For today)

- Scheduling Algorithms
  - Rate Monotonic Algorithm (RMA)
    - Optimal static priority, periodic task scheme
  - Earliest Deadline First (EDF)
    - Optimal dynamic priority, periodic task scheme
  - Maximum Urgency First (MUF)
    - Predictable Dynamic Priority
  - Deferred Server (DS)
    - Approach to handling sporadic tasks

- RTOS Overhead

- Synchronization and Communication
Critical Regions

Area of code accessing serially reusable resources

• Collision - Simultaneous use of a single, serially reusable resource

• Semaphore - a lock that protects critical regions

Mutual Exclusion mechanism required
Why Not Just Use a Software Flag?

```c
while (critical_region_in_use);
critical_region_in_use = TRUE;

use_critical_region();

critical_region_in_use = FALSE;
```
Semaphores

- Special memory location that acts as a lock
- Semaphore Primitives
  - wait - P(S)
  - signal - V(S)
- Must Be Atomic
  - Test-and-set instruction
  - Lock out interrupts or task swapping

Binary Semaphore [PL] p. 176
• Protect a Set of Resources

• Wait (assumed atomic)
  
  ```c
  int S1 = MAX_COUNT-1; /* init semaphore */
  void MP(int *S)
  {
     while (*S < 0); /* “spin lock” */
     *S -= 1;
  }
  ```

• Signal (assumed atomic)
  
  ```c
  void MV(int *S)
  {
     *S += 1;
  }
  ```

• Preferred Method is to Suspend and Resume
Priority Inversion (sophisticated case) [JC] 9.3.3

- Tasks T1, T2, T3 in Priority Order
- Tasks T1 and T3 Access Same Shared Resource

![Diagram showing priority inversion](image-url)

T3
T1
T2

T1 preempts T3
T1 suspends
T2 preempts T3
T1 preempts T3 again

(Spin lock causes deadlock)

• Solution: Prevent Task Swapping During Lock
• Better Solution: Bump Up Priority
• **Priority Inheritance**: (T1 > T2 > T3)
  • T1 Blocks on a Locked Semaphore (suspends)
  • If T1 is Blocked on T2’s Lock, T2’s Priority = T1’s
  • When T2 Unlocks, T2 Priority Reverts
• Priority Inheritance is Transitive
  • T3 Blocks T2 which Blocks T1; T3 Inherits Priority of T1 through T2
  • Tx can Preempt Ty if Tx not Blocked and Current Priority of Tx > Current Priority of Ty
Priority Inheritance

- Tasks T1, T2, T3 in Priority Order
- Tasks T1 and T3 Access Same Shared Resource

T1 preempts T3
T1 blocks
T2 releases (T3 now has T1 priority)
T1 preempts T3 again

T3 inherits T1 priority
T2
T3 unlocks SR
reverts priority

- Priority Inheritance can still Deadlock
  - T1 > T2
  - T1: P(S1); P(S2); V(S2); V(S1); (properly nested)
  - T2: P(S2); P(S1); V(S1); V(S2); (properly nested)
  - If T2 starts first and T1 preempts before P(S1) :-(

- Priority Ceiling Protocol
  - Priority Ceiling is Highest Priority of Any Task that may Lock It
  - T1 Blocks when Any Semaphore with Equal or Higher Ceiling is Locked
Priority Ceiling Protocol

- Tasks $T_1$, $T_2$, $T_3$ in Priority Order
- Tasks $T_1$ and $T_3$ Access Same Shared Resources
Mailboxes - Communication and Synchronization

- Known Memory Location with Key
  - Post - write to the location
  - Pend - read from the location

- Scheduler allows Post and Pend operations

- If no key in the mailbox, the task **blocks**
  - Accept or Check operation - don’t block if no key

- Scheduler checks for keys and unblocks tasks

- Mailbox Interrupt may force this (no busy waiting)

- Mailbox queues (pending requests)
Communication Methods

• Global Variables
• Double Buffers
  • Time-Relative Data (correlated)
• Triple Buffers (Chimera)
• Ring Buffers (Circular Queue)
• Message Passing
Message Passing

- Setup Message Queues for message packets
- Use Mailboxes
- Secure Point-to-Point Communication
  - One-to-One
  - One-to-Many
  - Many-to-One
- Often Implemented as UNIX-Style Sockets
- Fairly High Overhead

Chimera: msgCreate(), msgAttach(), msgDetach(), msgSend(), msgReceive(), msgKill()
Chimera State Variables

- Global Variables
- Provably Optimum Communication Performance
- Automatic Multiprocessor Address Resolution
- No Operational Overhead
  - Overhead only during initialization
- Point-to-Point Communication
  - One-to-One
  - One-to-Many
- Many-to-One (Join Connector) Implemented as Module
Chimera Block Diagrams

• Simple Processor-Inspecific Construction of Apps
• Amenable to Visual Programming
Graphical Programming with Chimera - STD’s

create

start

initHand1

Preshape&Move1

Grasp1

ViaMove1

GuardedMove1

Ungrasp1

initHand2

Preshape&Move2

Grasp2

ViaMove2

GuardedMove2

Ungrasp2

ViaMove3

end

destroy

reset

halt
No applications are currently executing.

Applications: trajstart.app1
Virtual Programming Interface
Virtual Programming Interface
ControlShell™ Block Diagrams

Real-Time Innovations, Inc. (www.rti.com)
StethoScope™ Real-Time Data Logging

Real-Time Innovations, Inc. (www.rti.com)
System Optimization

- Compute Tasks at Slowest Cycle Possible
  - *Ex: Helicopter cabin temperature*

- Use Scaled Integer Arithmetic
  - Integer Calculations Faster Than Floating Point
  - Mostly I/O-Based with Finite Precision

- Use Look-Up Tables for Complex Functions
  - Very Slow Transcendental Functions: Sine, Cosine, Tangent
  - Application-Dependent Mappings
  - Non-Linear Functions
Code Optimization

• Eliminate Arithmetic Identities
  \[ a + 0, b \times 1 \]

• Reduction in Strength
  Use fastest instruction per task

• Common Subexpression Elimination
  \[
  \begin{align*}
  x &= y + a \times b; \\
  t &= a \times b; \\
  y &= a \times b + z; \\
  x &= y + t; \\
  y &= t + z;
  \end{align*}
  \]

• Use of Registers and Caches
  see above

• Intrinsic Functions
  in-line code and macros