Comments on RTAI & pipes, named pipes, FIFOS
Lecture 11

RT extension to Linux
L11: Critical Section Synchronization

Task T0 – Producer
while(1)
    If not full
        add item
        increment count
    else
        wait for space
end while

Task T1 – Consumer
while(1)
    If not empty
        get item
        decrement count
    else
        wait for item
end while

Figure 12.27 State Chart Diagram Modeling a Producer-Consumer Information Exchange
L11: Critical Section Synchronization

• Critical Section – resource shared by tasks, reading and writing common variables
• Mutual Exclusion Synchronization (Block)
• Condition Synchronization (delay until true/false)
• Deadlock (chapter 13)
• Atomic Operations (indivisible)

While(1)
  non critical code
  entry section
  critical section
  exit section
  non critical code
End while

“Announces to all”
Read about Flags (12.7.2) and Token Passing (12.7.3)

**Interruption**
- Problem happens when preemption is allowed
  → temporarily disable it.

While(1)
  non critical code
  **entry section**
  critical section
  **exit section**
  non critical code
End while

Disable interrupts
Enable interrupts
Semaphores

- Binary semaphores
- Counting semaphores

Atomic operations: wait & signal (post)

Task T0:

```c
... wait(s) {
    while(s);
    s = TRUE;
}
s initialized to FALSE
```

Task T1:

```c
... signal(s) {
    s = FALSE;
}
```

(Binary case)
L11: Critical Section Synchronization

```c
/* Copyright (C) 2005-2014 ViGIR Vision-Guided and Intelligent Robotics Lab
 * Written by Guilherme DeSouza, Luis Rivera
 * This program is free software; you can redistribute it and/or modify it under
 * the terms of the GNU General Public License as published by the Free Software
 * Foundation, meaning: keep this copyright notice, do not try to make money out
 * of it, it's distributed WITHOUT ANY WARRANTY, yada yada yada... */

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <semaphore.h>
#include <stdlib.h>

void My_thread1(void *ptr);
int counter = 0;
sem_t my_semaphore;

int main(void)
{
    pthread_t thread1;
    srand(time(NULL)); // initialize random seed
    // initialize semaphore with value 1 (what if initialized with 0?)
    sem_init(&my_semaphore, 0, 1);
    pthread_create(&thread1, NULL, (void*) &My_thread1, NULL);
    while(counter < 1000)
    {
        usleep(500000+rand()%1000000);
        sem_wait(&my_semaphore);
        counter++;
        sem_post(&my_semaphore);
        printf("My_thread1: %d \n", counter);
    }
    pthread_exit(0);
}

/test_third_semaphore
My_thread1: 1
My_thread1: 2
ParentThread: 3
ParentThread: 4
My_thread1: 5
My_thread1: 6
ParentThread: 7
My_thread1: 8
ParentThread: 9
My_thread1: 10
ParentThread: 11
ParentThread: 12
My_thread1: 13
ParentThread: 14
My_thread1: 15
ParentThread: 16
My_thread1: 17
ParentThread: 18
My_thread1: 19
ParentThread: 20
My_thread1: 21
```
Assignment

• Read section 12.7
• Readings on semaphores, mutexes and related concepts. Links on Blackboard: Course Content > Other Material

• Study for Mid-term
  – Book chapters: 11, 12
  – Lecture notes: 1 - 11
  – Assignments: HW 1; reading assignments, including links and documents posted on Blackboard
  – Labs: 1, 2, 3

No textbook or lecture notes allowed. You can bring calculator, ruler, pen, pencil, eraser.