Notes in ECEN 5623

Zachary Vogel

March 4, 2016

stuff

read 8, exercise due on 3/5, exam on Tuesday. ITLL tour at end of class on Thursday.
exam will be closed book.
3/15 form groups for Exercises 5 and 6 and final project.

Lecture

known terms:
RTL (register transfer logic), API, Cyclic executive(superloop), atomic operation(disable interrupt, executes, enables interrupts), best effort, binary semaphore, blocking, BSP(board support package), cache hit, canonical service(standard way of writing a service or task), Completion test, context switch, CPI, critical instant, critical section, (D,T anc C)(Deadline, time and computation), Deadline, dispatch, DMA(deadline monotonic access, or direcct memory access), EDF, LLF, FCFS, Feasibility Test, FIFO, Fixed-Priority, Hard Real-time, Harmonic, Isochronal, Jiffy, Jitter, Laxity, LCM, LLF, Livelock, Main+ISR, memory-mapped IO, Message queue, mutex semaphore (specifically for locking), necessary and sufficient, Period transform(altering task period to make RM work), POSIX, preemption, Priority, priority ceiling, priority inversion, rate-monotonic, real-time, Rate monotonic analysis (RMA), scheduling point, Semaphore, Service, Shared memory, soft real-time, task, tick, time-slice, timeout, utility curve, wcet

Real-time correctness, how to implement RM, what are harmonic service sets, scheduling feasibility test, interference and blocking, utility curves, difference between ceiling protocal and priority inheritance, RMA schedulability test formulation, POSIX RT pthreads and sync, LLF and EDF scheduling policies.

be able to:
define and draw utility curves
derive the RM LUB (be able to answer questions about it)
analyze blocking code, including deadlocks
describe a scheduler state machine using POSIX API calls
draw timing diagrams of multi-service systems
derive the DMA schedulability test
Draw timing diagrams to prove feasibility of a service set using RMA, DMA, EDF, or LLF

- CH 1: introduction, RT correctness, definitions
- CH 2: system resources, CPU, I/O, memory bound
- CH 3
- CH 4
- CH 5

- CH 6
- Cheddar potentially

know a scheduling statemachine diagram.
ready state= only need CPU
i/o or shared memory= pending

Fundamentals of RT analysis:

- RT correctness= before deadline and correct result
- utility curves for best effort, Hard RT, isochronal RT, and soft RT
- CPU, I/O and memory resource space
- Basic Timing Diagrams
- Theorem 1- RM Least Upper Bound
- Theorem 2 (Lehoczky, Shah, Ding) - if deadlines are met over longest period(or better yet, LCM) from C.I. then system is feasible
- fixed prioirty, preemptive, run-to-completion scheduling
- sufficiency and necessary condition types

deadline monotonic theory

- differences between this and RM, T$\neq$D, prioirty assignmetn policy, iterative feasibility test
- DM priority assignmetn policy
- simple sufficient feasibility test
- improved (more necessary) feasibility test

RM theory, C=WCET, T=D, critical instant
feasbility test, derivation of 2 task sufficient LUB
scheduling point - O(n^3), but N& S, same with completion one.
dynamic priority theories, when to use them

Linux and POSIX RT extensions

- pthread_create and join
- SCHED_FIFO, priorities RT Max and min, attributes
- SCHED_OTHER
- CPU affinity
- real time clock (relative vs absolute time)
- blocking and timeouts, timespec struct, timed wait
- logMsg versus printf, printf i/o deeelays caller and can't be called in kernel/ISR context
- logMsg performs output in slack time via tlogTask and message queue interface

RT sync

- priority inversion

- unbounded priority inversion
- priority inheritance
- priority ceiling
- necessary 3 conditions for unbounded prio inversion
  - 3 or more tasks
  - H and L tasks involved in mutex
  - 1 or more M tasks not involved in mutex cause interference
- Mars pathfinder story
  - what went wrong
  - why
  - how was it fixed
  - priority inversion happened
  - fixed with patch which did a type of priority inheritance

LINUX
POSIX RT Extensions

- Message queues
  - Priority enqueue and dequeue
  - Same priority
  - blocking vs non-blocking send and receive
- REal-time signals won't be questions on this
  - signals that queue -why?
  - passing data- how?
- real-time interval timers and clocks

Service efficiency concepts
blocking is evil
path length
path execution efficiency
intermediate I/O
overlapping intermediate I/O with CPU

CODE WRITTEN ON EXAM NEED NOT COMPILE