Embedded Operating Systems

Che-Wei Chang

chewei@mail.cgu.edu.tw

Department of Computer Science and Information Engineering, Chang Gung University

An Real-Time OS: µC/OS-II Quick Overview

Introduction of µC/OS-II (1/2)

- The name is from micro-controller operating system, version 2
- µC/OS-II is certified in an avionics product by FAA in July 2000 and is also used in the Mars Curiosity Rover
- It is a very small real-time kernel
  - Memory footprint is about 20KB for a fully functional kernel
  - Source code is about 5,500 lines, mostly in ANSI C
  - It's source is open but not free for commercial usages
- Preemptible priority-driven real-time scheduling
  - 64 priority levels (max 64 tasks)
  - 8 reserved for µC/OS-II
  - Each task is an infinite loop

Introduction of µC/OS-II (2/2)

- Deterministic execution times for most µC/OS-II functions and services
- Nested interrupts could go up to 256 levels
- Supports of various 8-bit to 64-bit platforms: x86, ARM, MIPS, 8051, etc.
- Easy for development: Borland C++ compiler and DOS (optional)
- However, uC/OS-II still lacks of the following features:
  - Resource synchronization protocol
  - Soft-real-time support

The µC/OS-II File Structure

Application Code (Your Code!)

Processor Independent Implementations

- Scheduling policy
- Event flags
- Semaphores
- Mailboxes
- Event queues
- Task management
- Time management
- Memory management

Application Specific Configurations

- OS_CFG.H
- Max # of tasks
- Max Queue length
- …

uC/OS-II Port for Processor Specific Codes

Software

Hardware

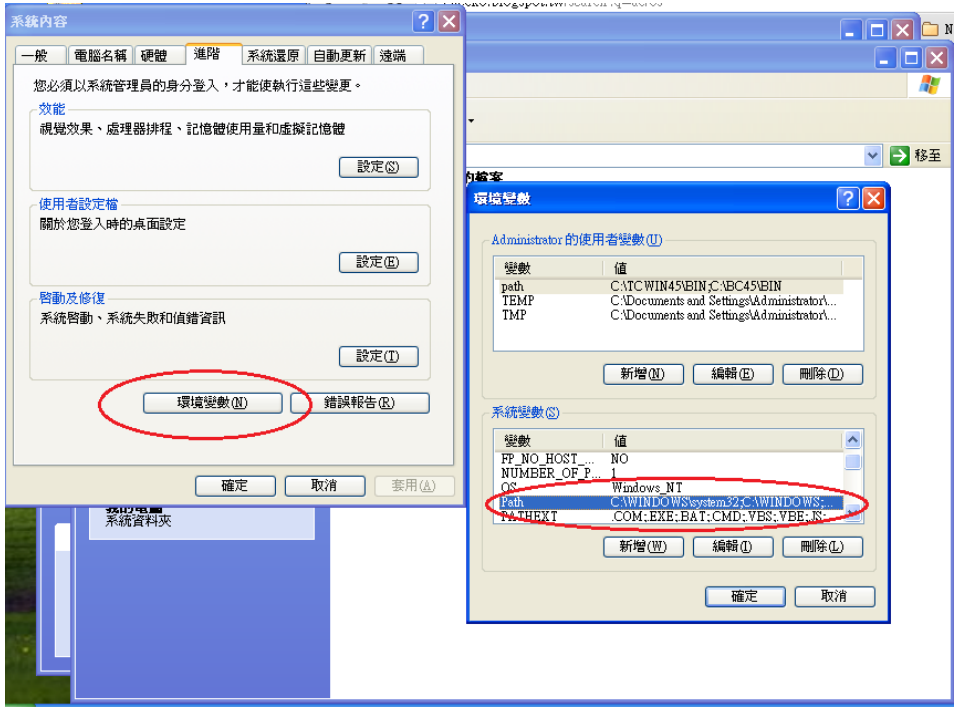
CPU | Timer

Requirements of µC/OS-II Emulator

- Operating System
  - Windows XP 32bits
  - Use virtual machine to install the OS
  - Install “Guest Additions” for Virtualbox
- Tools
  - Borland C++ compiler (V4.5)
    - BC45 is the compiler
  - Turbo Assembler
    - The assembler is in tasm
  - The source code and the emulation environment of µC/OS-II
    - SOFTWARE is the package

Borland C++ Compiler

- Download Borland C++ and install it on your windows XP environment
  - Double click the “INSTALL.EXE”
- Add “;C:\BC45\BIN” to your system Path



Turbo Assembler

- Download Turbo assembler and unzip the file
- Copy “\tasm\BIN\TASM.EXE” to your “C:\BC45\BIN”
  - Include the missing assembler which is going to be used during we compile the source code of µC/OS-II

Compile µC/OS-II Example Code

- Download the source code and emulator µC/OS-II
  - It is recommended to put the source code package "SOFTWARE" directly in C:\
- Test the first example
  - Execute C:\SOFTWARE\uCOS-II\EX1_x86L\BC45\TEST\TEST.EXE
  - Press ECS to leave
- Rename or remove the executable file
  - Rename TEST.EXE
- Compile the µC/OS-II and the source code of the first example
  - Run C:\SOFTWARE\uCOS-II\EX1_x86L\BC45\TEST\MAKETEST.BAT
  - A new "TEST.EXE" will be created if we compile it successfully

Common Mistakes

- Did you directly put the package “SOFTWARE” in C:\ ?
- Have you copied the correct file “TASM.EXE” to your “C:\BC45\BIN” directory?
- Did you set the Path correctly?
  - See the picture in Page 7
  - There is no space

Project Requirements

CPU Scheduler

- Short-term scheduler selects a process among the processes in the ready queue, and allocates the CPU to the selected process
  - Queue may be ordered in various ways
- CPU scheduling decisions may take place when a process:
  1. Switches from running to waiting state
  2. Switches from running to ready state
  3. Switches from waiting to ready
  4. Terminates
- Scheduling under 1 and 4 is nonpreemptive
- All other scheduling is preemptive

Dispatcher

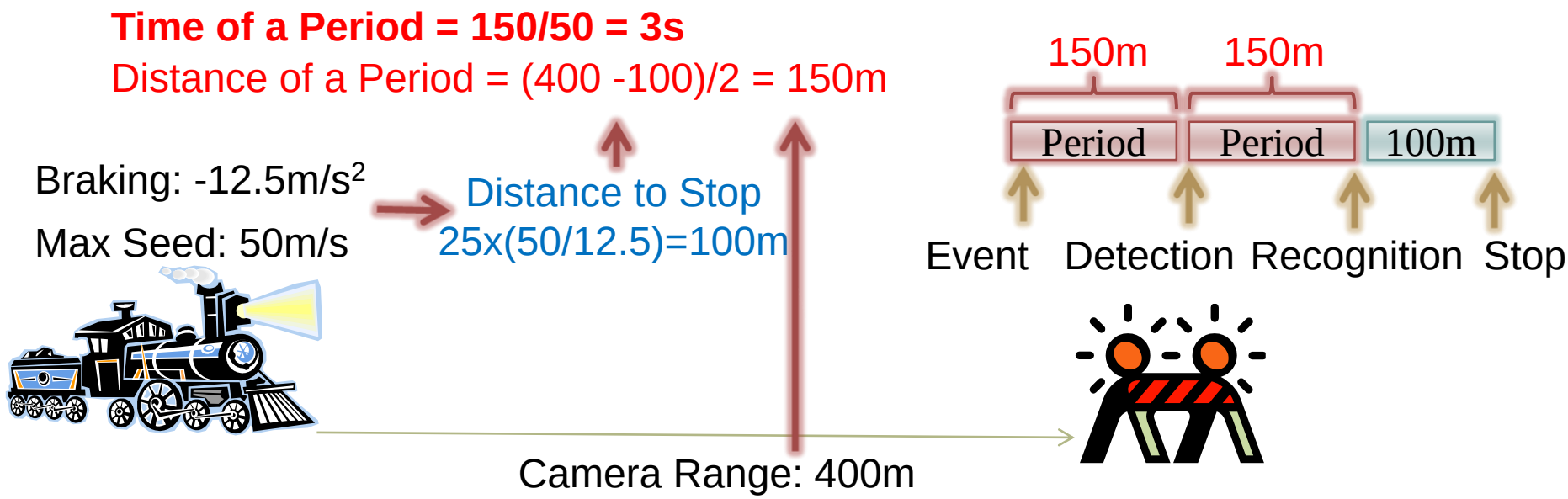
- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler
  - switching context
  - switching to user mode
  - jumping to the proper location in the user program to resume that process
- Dispatch latency – the time it takes for the dispatcher to stop one process and start another running

Scheduling Algorithms

- First-Come, First-Served Scheduling (FIFO)
- Shortest-Job-First Scheduling (SJF)
- Priority Scheduling
- Round-Robin Scheduling (RR)
- Multilevel Queue Scheduling
- Multilevel Feedback Queue Scheduling
- Multiple-Processor Scheduling

An Example of Real-Time Tasks

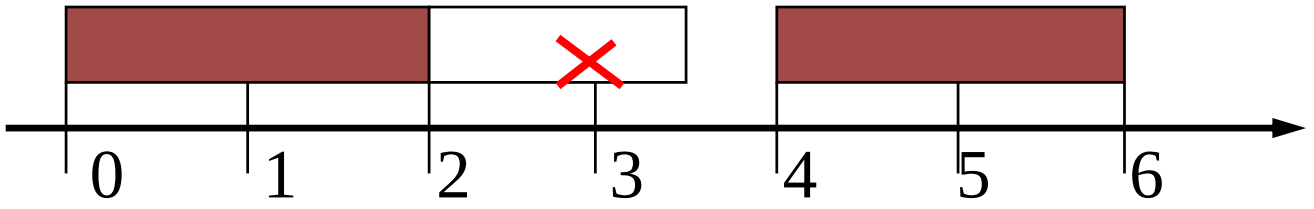
- A camera periodically takes a photo
- The image recognition result will be produced before the next period
- If there is an obstacle, the train automatically brakes



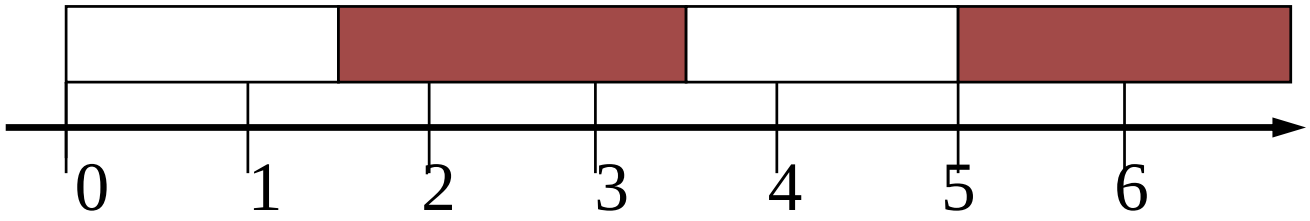
Periodic Task Scheduling

- Studying: 2 days per 4 days 
  Playing Basketball: 1.5 days per 3 days 

- Case 1: Studying is always more important



- Case 2: Doing whatever is more urgent



Project Requirements

- Read the input file and create the periodic tasks
- Implement the rate monotonic (RM) scheduler
- Implement the priority inheritance protocol (PIP)
- Bonus 1 (10%): Implement the priority ceiling protocol (PCP)
- Bonus 2 (10%): Implement the earliest deadline first (EDF) and let it work with PCP

Input File Format

- At most 7 tasks
- At most 7 semaphores
- The greatest common divisor (GCD) of the periods of all tasks is less than 3000 seconds
- Input file format: (all are integers)

N

$S_1\ T_1\ E_1\ I_{1,1}\ L_{1,1}\ R_{1,1}\ \ldots I_{1,\,s1}\ L_{1,s1}\ R_{1,s1}$

$\vdots$

$S_N\ T_N\ E_N\ I_{N,1}\ L_{N,1}\ R_{N,1}\ \ldots I_{N,sN}\ L_{N,sN}\ R_{N,sN}$

- N: Number of periodic tasks
- S_x: Number of semaphores used by task i
- T_x: Period of task i
- E_x: Execution time of task i
- $I_{x,y}$: The index number of the y-th semaphore of x-th task
- $L_{x,y}$: The request time of the y-th semaphore of x-th task
- $R_{x,y}$: The release time of the y-th semaphore of x-th task

Example of Input 1 (unit:sec)

3

2 20 5 0 1 2 1 2 4

1 25 5 2 3 5

2 30 11 2 0 2 1 9 11

Example of Input 2 (unit:sec)

4

1 20 4 0 1 2

1 24 3 1 0 2

1 28 4 2 1 3

2 32 12 2 0 4 0 8 12

Needed Files

- Source code of your project (SOFTWARE)
- Report: 4 pages
- Deadline: 20:00 on 2023/12/18
- Upload to the e-learning system
- The grading baseline: 90

File Formats

- File name: EOS-Project-StudentID-Report
- File type: PDF
- File name: EOS-Project-StudentID-Source
- File type: ZIP