Operating System & Networks 2003

Uwe R. Zimmer – International University Bremen

what is offered here?

Overviews, Paths, Definitions, Terminology,
Foundations, Methods, Algorithms
Realities,
Current research trends, Projects,
Perspectives,
… and some theory

into/for/about *Operating Systems & Networks*

who could be interested in this?

anybody who …

… would like to see
how rich, diverse and deep the real world of operating systems goes

… would like to learn how to create predictability
and fault-tolerant operating systems

… would like to know more about the usage of
95% of all µprocessors (and thus operating systems)

who are these people? – introduction

This course will be given by

Holger Kenn for the networks sections

and

Uwe R. Zimmer for the operating systems sections

how will this all be done?

☞ Lectures (320-202):

- 2 per week … all the nice stuff and theory
  Tuesday, 8:00-9:15; Friday, 11-12:15 – all in Conrad Naber lecture hall

☞ Labs (Advanced CS lab), independent course, but related (320-222):

- 2 sessions per week … all the rough stuff and practice
  Monday 15:30-19:30; Tuesday 15:30-19:30

☞ Resources:

- introduced in the lectures and collected on the course page:
  `http://www.faculty.iu-bremen.de/course/FundCS2/`
  … as well as schedules, slides, code, etc. pp. … keep an eye on these pages!

☞ Assessment:

- Two exams, 50% each, one oral exam, one written exam – assignments for self-checking

Topics in operating systems

1. Introduction
2. Hardware basics
3. Processes
4. Memory management

Table of contents

2. Hardware Fundamentals

- **General computer architecture**
- **CPU**
    - Registers
    - Traps/Interrupts & protected modes
- **Memory**
    - General memory layout
    - Caching
- **I/O systems**
    - I/O controllers, I/O buses, device programming
- **Some examples of µprocessors**
    - Small scale µcontroller (68HC05)
    - Full scale integrated processor (MCP565)

Table of contents

3. Processes

- **Processes and threads**
  - Architectures, definitions, process states
- **Synchronization**
  - Shared memory based synchronization
  - Message based synchronization
- **Deadlocks**
  - Detection, avoidance, and prevention (& recovery)
- **Scheduling**
  - Basic performance based scheduling
  - Basic predictable scheduling
  - Aperiodic, sporadic, and synchronized tasks

Table of contents

3.1 Synchronization methods

- **Shared memory based synchronization**

• Semaphores	☞ ‘C’, POSIX — Dijkstra
• Conditional critical regions	☞ Edison (experimental)
• Monitors	☞ Modula-1, Mesa — Dijkstra, Hoare, …
• Mutexes & conditional variables	☞ POSIX
• Synchronized methods	☞ Real-time Java
• Protected objects	☞ Ada95

- **Message based synchronization**

• Asynchronous messages	☞ e.g. POSIX, …
• Synchronous messages	☞ e.g. Ada95, CHILL, Occam2
• Remote invocation, remote procedure call	☞ e.g. Ada95, …
• Synchronization in distributed systems	☞ e.g. CORBA, …

3.2 Deadlocks

- **Ignorance & recovery**
  - ☞ ‘kill some seemingly persistently blocked processes from time to time’ (exasperation)
- **Deadlock detection & recovery**
  - ☞ multiple methods for detection, e.g. resource allocation graphs, Banker’s algorithm
  - ☞ recovery is mostly ‘ugly’
- **Deadlock avoidance**
  - ☞ check system safety before allocating resources, e.g. Banker’s algorithm
- **Deadlock prevention**
  - ☞ eliminate one of the pre-conditions for deadlocks

3.3 Scheduling

- **Basic performance based scheduling**
    - C_i *is not known*: first-come-first-served (FCFS), round robin (RR), and feedback-scheduling
    - C_i *is known*: shortest job first (SJF), highest response ration first (HRRF), shortest remaining time first (SRTF)-scheduling
- **Basic predictable scheduling**
    - Fixed Priority Scheduling (FPS) with Rate Monotonic (RMPO)
    - Earliest Deadline First (EDF)
- **Real-world extensions**
    - Aperiodic, sporadic, soft real-time tasks
    - Synchronized talks (priority inheritance, priority ceiling protocols)

Table of contents

4. Memory