Operating Systems and Concurrency

Lecture 1: Introduction

COMP2007

Geert De Maere
(Dan Marsden)
{Geert.DeMaere, Dan.Marsden}@Nottingham.ac.uk

University Of Nottingham
United Kingdom

2023

Goals

What Will be Covered?

- Introduction to the **fundamental concepts**, **key principles** and **internals** of (old and new) **operating systems** and **concurrency**
- Better understand how **application programs interact with the operating system**
- Basic understanding of writing **concurrent** / **parallel code** and **OS principles** related to concurrency

Lectures

When and Where?

- Through **lectures** on Mon, Wed, and Thu (4 weeks)
- Recordings will be available (will try to live stream over Echo360)
- Remember, studying is also about **interaction with peers** and **building contacts for life**

Lectures

Subjects We Will Discuss

Subject	#Lectures	By
Introduction to operating systems/computer design	3	GDM/DM
Processes, process scheduling, threading, ...	4	DM
Concurrency (deadlocks)	5	DM
Concurrency/coursework clarification/revision	1	GDM/DM
Memory management, swapping, virtual memory, ...	6	GDM
File Systems, file structures, management, ...	5	GDM
(Virtualisation & Cloud)	2	GDM
Revision	1	GDM

Table: Preliminary course structure

Labs

What and How?

- Labs are on Fridays (09:00 - 10:00) in A32, CS (from W/C 9th of October)
- The **labs** will teach you:
  - **OS concepts** (processes, schedulers, shared memory)
  - The use of operating system APIs & **implementation** / **coding** on Linux systems
  - The basics of **concurrency**
- **Lectures** will **introduce** these concepts

Coursework

Content

- The coursework focuses of a **OS structures**, **process scheduling**, **concurrency**, and **threads** (not **processes** / **fork**)
  - **Draft specification** will be available W/C 9th of October on Moodle (read this ASAP)
  - Follow the guidance to **break it down** in steps!
- It requires **C programming**

Submission

- The recommended submission date is the **12th of December** (latest date is **04/01/2024**)
- NO late submissions (unless you have ECs)!

Reading Material

My Favourite Books

- Seminal books:
    - *Tanenbaum, Andrew S. 2014 Modern Operating Systems. 4th ed. Prentice Hall Press, Upper Saddle River, NJ, USA.*
    - Silberschatz, Abraham, Peter Baer Galvin, Greg Gagne. 2008. *Operating System Concepts*. 8th ed. Wiley Publishing.
    - Stallings, William. 2008. *Operating Systems: Internals and Design Principles*. 6th ed. Prentice Hall Press, Upper Saddle River, NJ, USA.
- Other sources:
    - Daniel P. Bovet, Marco Cesati *Understanding the Linux Kernel*. 3rd ed. O'Reilly Media, November 2005
    - Slides and recordings will be available on Moodle

Assessment

Exam & labs

- **In person ExamSys** (2 hours - TBC) that focuses on **knowledge**, **understanding**, **application**
  - The exam will have **3 out of 4 questions**, with 50% of the assessment on the exam
  - **Sample questions** from previous years are available on Moodle and are included in the lectures (**answers** are not available)
- Labs are part of the exam:
  - One or more (partial) questions in the **exam will be designed to evaluate the labs**
  - Help you with some aspects of the coursework (e.g. coding systems)

Assessment

- The coursework is an **individual** task and counts for 50%
  - **Git repositories** have been set up for you
  - **Only the final version** should be submitted in Moodle
  - **Submit** your code **regularly** (Git and Moodle- as many times as you like)
- **Academic misconduct** will be followed up on!

Assessment

Workload

- This is a **20 credit module** - **200 hours** of work (5 × 40 hour week)
- The **coursework** should take approximately 100 hours
- **Lectures** take approximately 24 hours, **labs** 9 hours
- 67 hours of **revision**, approx 3 hours per **lecture**, 8 minutes per **slide**

Assessment

Start Revision Early

An E-mail Received Evening Before the Exam

Hi Geert,

*There is a **lot of information** covered during the course, hence **making revision very challenging**.*

*Do you have any guidance as to how to break the course down in to a **list of "main topics"** that are **essential to know in detail**?*

*Also, how will these **topics be split in to the 5 optional questions** in the exam?*

Thanks

…

Assessment

Start Revision Early

Response

Dear . . . ,

Unfortunately, I am ***unable to provide any information*** *other than what was said during the lectures: the exam will try (as much as possible) to* ***assess all aspects covered in the module****. This is the only way in which a* ***fair exam*** *could be put together, since different students will find different topics easier/more difficult.*

This is probably not the answer that you were hoping for, but if I would give ***you a more detailed answer****, it* ***may be unfair to other students*** *who were not provided with this information.*

Best wishes,

Geert De Maere

About Us

Contact Details

- GDM's Contact details:
  - Name: Geert De Maere
  - E-mail: Geert.DeMaere@Nottingham.ac.uk
  - Office: C84
  - Office hour: Tuesdays 14:00 - 15:00 (confirm attendance by e-mail)
- DM's Contact details:
  - Name: Dan Marsden
  - Office: C41
  - E-mail: Dan.Marsden@Nottingham.ac.uk

About Me

Background

- Graduated in 2000, Bsc, Msc in Engineering
- Completed my PhD in CS in 2010 (Operational Research)
- Specific interest in **airline scheduling**, **airport operations** and **energy**
- I work together with **Institute for Aerospace Technology** (IAT), NATS, Heathrow Airport, etc.

About Me

My Background

- How does my research link in with operating systems?
  - I work on **scheduling** and **optimisation**
  - Exploit computer **architecture/design** and **common principles** in operating system design to:
    - Implement **sensible** parallelisations of algorithms
    - Speed up algorithms (caching, manipulate registers)
  - **Exploit similar principles** and **work on similar problems** in my daily work (e.g. caching, parallelisation, machine scheduling with sequence dependent setup times)
  - ...
- *“The ability to think independently while giving due weight to the arguments of others”*

Defining Operating Systems

What Can an OS Do For Me?

```java
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

public class Demo1 {
  public static void main(String[] args) throws IOException {
    FileWriter fw =
      new FileWriter(("C:/Program Files (x86)/test.txt"));
    PrintWriter pw = new PrintWriter(fw);
    pw.close();
  }
}
```

Defining Operating Systems

What Can an OS Do For Me?

- **File systems**: **where** is the file physically written on the disk and how is it **retrieved**?
- **Abstraction**: why looks the instruction the same **independent of the device**?
- **Concurrency**: what if multiple programs **access the same file simultaneously**?
- **Security**: why is the **access denied**?

Defining Operating Systems

What Can an OS Do For Me?

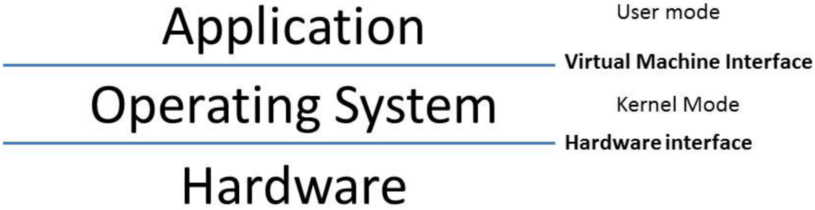
```java
public class Demo2 {
  public static void main(String[] args){
    long[] aLargeArrayOfLargeNumbers
                = new long[Integer.MAX_VALUE];
  }
}
```

- **Where** in memory will the array be stored and how is it **protected** from unauthorised access?
- What if the array requires **more memory than physically available**?
- What if only **part of the array is currently in use** ?
- What if an **other process starts running**?

Defining Operating Systems

A Virtual Machine Providing Abstractions

- In the early days, programmers had to **deal directly with the hardware**
  - Real computer **hardware is ugly**
  - Hardware is **extremely difficult** to manipulate/program
- An operating system is a layer of indirection on top of the hardware:
  - It provide **abstractions** for application programs (e.g., file systems)
  - It provides a **cleaner and easier interface to the hardware** and hides the complexity of **“bare metal”**
  - It allows the programmer to be lazy by using **common routines** :-)



Application

User mode

Virtual Machine Interface

Operating System

Kernel Mode

Hardware interface

Hardware

Defining Operating Systems

Some Wisdom

David Wheeler (First PhD in Computer Science, 1951)

> *“All problems in computer science can be solved by another level of indirection”*

Defining Operating Systems

A Resource Manager

- Many modern operating systems use **multi-programming** to **improve user experience** and **maximise resource utilisation**
  - Disks are slow: without multi-programming, CPU time is wasted while waiting for I/O requests
    - Imagine a **CPU** running at 3.2 GHz (approx. 3.2×10^9 instructions per second)
    - Imagine a **disk** rotating at 7200 RPM, taking 4.2 ms to rotate half a track
    - I/O is slow, we are **missing out on** $3.2 \times 4.2 \times 10^6$ **instructions** (13.44m)!
- The implementation of **multi-programming** has important **consequences** for **operating system design**

Defining Operating Systems

A Resource Manager

- The operating system must **allocate/share** resources (including CPU, memory, I/O devices) **fairly** and **safely** between **competing processes**:
  - In time, e.g. CPUs and printers
  - In space, e.g., memory and disks
- The execution of **multiple programs** (processes) needs to be **interleaved** with one another:
  - This requires **context switches** and **process scheduling** ⇒ **mutual exclusion**, **deadlock avoidance**, **protection**, . . .

Summary

Take-Home Message

- Summary:
    - Structure of the **module & assessment**
    - Introduction to **operating systems**
    - Operating systems in terms of **abstractions** and **resource managers**
- Tasks:
    - Revise your knowledge of C