University of BRISTOL

Computer Systems B

COMS20012

Introduction to Operating Systems and Security

What is an OS?

What is an operating system?

- A computer program that
  - **Multiplexes** hardware resources
  - Implements resource **abstractions**

What is an operating system?

- A computer program that
  - **Multiplexes** hardware resources
  - Implements resource **abstractions**

It is just another program…

What is an operating system?

- A computer program that
  - **Multiplexes** hardware resources
  - Implements resource **abstractions**

It is just another program… but a large and complex one

- most complex piece of code you would have seen so far

What is an operating system?

- A computer program that
  - **Multiplexes** hardware resources
  - Implements resource **abstractions**

It is just another program… but a large and complex one

  - most complex piece of code you would have seen so far

- **Multiplexing:** allows multiple people or programs to use the same set of hardware resources—processors, memory, disks, network connection—safely and efficiently.
- **Abstractions:** processes, threads, address spaces, files, and sockets—simplify the usage of hardware resources by organizing information or implementing new capabilities.

University of BRISTOL

Why study OS?

Why study OS?

- **Reality:** this is how computers really work, and as a computer scientist or engineer you should know how computers really work.

Why study OS?

- **Reality:** this is how computers really work, and as a computer scientist or engineer you should know how computers really work.
- **Ubiquity:** operating systems are everywhere, and you are likely to eventually encounter them or their limitations.

Why study OS?

- **Reality:** this is how computers really work, and as a computer scientist or engineer you should know how computers really work.
- **Ubiquity:** operating systems are everywhere, and you are likely to eventually encounter them or their limitations.
- **Beauty:** operating systems are examples of mature solutions to difficult design and engineering problems. Studying them will improve your ability to design and implement abstractions.

Information about the labs

bristol.ac.uk

Challenging labs

- Do look at the lab in advance
- Work in pair
  - It will be hard to complete the task alone
  - For lab 7 do review each other work within your lab “bubble”
- Labs are cummulative
- There are (many) tests to verify your solution work
- Be ready to work beyond the 3h lab timeslot

Challenging labs

- Do look at the lab in advance
- Work in pair
  - It will be hard to complete the task alone
  - For lab 7 do review each other work within your lab “bubble”
- Labs are cummulative
- There are (many) tests to verify your solution work
- Be ready to work beyond the 3h lab timeslot
- … you will need to **program in C**

Challenging labs

- Do look at the lab in advance
- Work in pair
  - It will be hard to complete the task alone
  - For lab 7 do review each other work within your lab “bubble”
- Labs are cummulative
- There are (many) tests to verify your solution work
- Be ready to work beyond the 3h lab timeslot
- … you will need to **program in C**
- Bring together things you learned in Year 1 and 2

Challenging labs

- Help each other!
- Many of you, very few of us
  - staffs (8 TAs)
- We do our best in labs but talking to each others help!

OS/161

- Instructional Operating System
  - Developed at Harvard (more info on Lab 5 page)
- Balance between realistic and mature systems (e.g., Linux) and instructional systems
- OS/161 runs in an emulator (sys161)
  - Emulates MIPS r2000/r3000 instruction set architecture
- sys161 simplify debugging and hardware support

What is in the labs?

- **Lab 5** (Week 18 – March 7)
  - Getting comfortable with the required tools
  - Learn to navigate OS/161 source code
  - Configuring and running your first kernel
- **Lab 6** (Week 20 – March 14)
  - Design and implements lock
- **Lab 7** (Week 22 – April 125)
  - Implement file-related system calls
  - Implement process-related systems calls
  - 3 weeks long lab
  - Start early, it is complex

University of BRISTOL

How to do well?

How to do well?

- Start the labs early (i.e. before the lab sessions)
- If you **finish a lab move to the next one**
- Work frequently and often
  - With your partner
  - With your lab “bubble”
- Make sure you attend all the lab sessions
- It is normal to find the lab hard…
  - … you will learn a lot

How to do well?

- Consider pair programming
  - Code together
  - Think things through, avoid bugs
- Use KASSERT
  - check what is an assertion if you don't know
- Iterate often and quickly
  - Do not write a lot of untested code
  - Small tested increment is the way to go
- Break your code in small functions!

University of BRISTOL

What to do now?

bristol.ac.uk

What to do now?

- Start going through Lab 5 **ASAP**!
- Get a development environment working.
- **Setup a git repository** to share your code with your partner.
- Find a partner!

You are Year 2 Computer Science, we expect a certain level of autonomy.

University of BRISTOL

Thank you

bristol.ac.uk