SOFTENG 370 Tutorial 1

Timo van Veenendaal

4 August 2020

Hello

I am Timo, you might remember me from last year. I am the tutor for 370 this semester.

- In these tutorials I'll cover some things about the previous week's content
- There should be some time for Q&A about assignments, past tests, etc. most of the time – I'll hang around at the end of the tutorial to answer questions
- Alternatively, email me at tvan508@aucklanduni.ac.nz to set a time
- If you have questions, ask on Piazza if possible instead of emailing so that the rest of the class can see!

Plan

Kernel

Question

What is kernel mode? How does it differ from user mode?

Kernel

Question

What is kernel mode? How does it differ from user mode?

- Having these two modes requires support from the hardware (i.e. the processor)
- When the processor is in kernel mode, all instructions can be executed, including **privileged instructions** which cannot be executed in user mode.
- What are some examples of privileged instructions?
- Why are privileged instructions (and kernel mode) needed?
- What happens if a program attempts to execute a privileged instruction in user mode?

Kernel

Question

How do ordinary programs get access to privileged instructions?

Kernel

Question

How do ordinary programs get access to privileged instructions?

- Through **system calls**

Kernel

Question

How do ordinary programs get access to privileged instructions?

- Through **system calls**
- A system call is a hardware mechanism that causes the processor to jump to a predefined location and enter kernel mode.
- This location will contain some special code that is part of the kernel to handle the call. The kernel can of course use privileged instructions to complete the call
- The kernel can run checks to make sure that the caller has permission to do what they want to do
- When the kernel is finished handling the call, it tells the processor to return to user mode and then returns execution to the caller

Linux

- You need access to Linux (or similar) for the assignments
  - You may also be able to get away with developing on macOS since it is also UNIX-based but you could run in to issues
- A few options
  1. Windows Subsystem for Linux
  2. Virtual machine
  3. FlexIT virtual machine
  4. Or dual boot...?
- Pick a distribution
  - I recommend either Ubuntu or Manjaro if you haven't used Linux much before
  - Otherwise use a distro of your choice
- Hopefully you are all comfortable with `bash` from 206

Windows Subsystem for Linux (WSL)

- The most convenient way to use Linux if your primary OS is Windows
- WSL 2, recently released, is essentially a wrapper around a Linux virtual machine. The older version of WSL is implemented differently
- Has excellent integration with VS Code through the “Remote – WSL” extension (you’ll want the C/C++ extension too)
- See `https://docs.microsoft.com/en-us/windows/wsl/install-win10` for installation instructions.
- Also check out the new Windows Terminal: `https://github.com/microsoft/terminal`
- Demo later

Virtual machine

- VirtualBox is free and good. A guide to installing Ubuntu on VirtualBox for Windows is available here: `https://brb.nci.nih.gov/seqtools/installUbuntu.html`
  - You might have used this in 206 last year
- Hyper-V is another alternative if using Windows Pro and is built in to the OS so you don't need to install anything

FlexIT

- Ubuntu VM in the cloud
- Requires installation of VMWare Horizon client (see instructions)
- Instructions at `https://www.auckland.ac.nz/en/students/my-tools/flex-it/flexit-guide.html`
    - Look for Ubuntu in the listing
- Demo

Demo

Demo of FlexIT and WSL now

The C Programming Language

- You might remember C from ENGGEN 131 (and maybe even COMPSYS 201)
- The way you will use C in this course is quite different to how it is approached in 131
  - Feels more ‘low-level’: heavy use of pointers, use of OS APIs (beyond just `printf`)...
- Compile on Linux using `gcc`

Compiling and gcc

- You might need to install gcc
  - Ubuntu: `sudo apt install build-essential`, you may have to do `sudo apt update` first
  - Equivalents exist for other distributions, Google if unsure
- To compile: `gcc -o <executable_name> sourcefile.c`
- Then to execute: `./executable_name`
- Let's do a demo

Questions

Any questions? About...

- The course in general?
- Installing Linux?
- Using C and gcc?

Otherwise see you all next week