Operating systems

Oscar Palm

SP 1 2025

Contents

Lecture 1

Introduction

1.1 Introduction to C

Code snippet 1.0

```
#include <stdio.h> // Tells the preprocessor to load a file

int main()       // The entrypoint of the program
{            // Defines a block of code, everything created in it and not returned is only defined for the block
   printf ("Hello world\n");  // A basic statement
  return 0;
}
```

For this course we will use Chalmers remote StuDat computers.

1.2 Course introduction

The course book is "Modern Operating Systems", by Tanenbaum and Bos.

Vincenso should work on his standup comedy.

1.2.1 Why study OS?

Provides services to system users. Studying it gives us an overview of how computers work on a lower level.

The operating system is both there to help the user actually use the computer, but also to protect the hardware from the main source of complications; the user. It makes sure that everything that needs to run on the computer is able to do so, and is given sufficient resources to perform its task.

1.3 What is an Operating system

An operating system is a program which provides a *set of services* to system users (human users and computer programs). It also protects the user from the hardware and vice verse.

An operating system manages resources on the computer and makes sure that all system users get the necessary resources (if possible), such as cpu(s), memory to store data, i/o devices and so on. It controls the execution of different programs, makes sure that an error in one doesn't cascade into others and makes sure different programs stay independent and cannot access each others' resources.

Saving a file

When the user is saving a file, without an operating system, nothing else would be able to happen on the computer while waiting for that; with a good operating system, the hardware resources can be managed in such a way that other tasks can be performed while waiting for the file to be written.

1.4 The event-driven (interrupt-drive) life of an OS

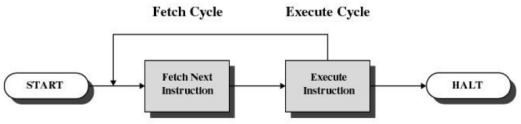

Figure 1.1: A simple fetch cycle where only one thing happens

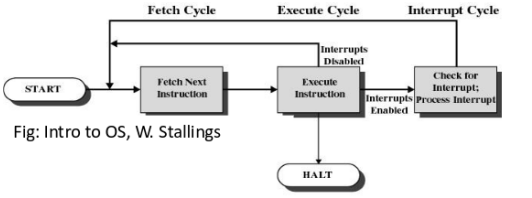

Figure 1.2: A more advanced fetch cycle where things can happen in the background

By looking for interrupts between running the instructions we are able to handle events from different parts of the system. With this we can achieve what was talked about in the example above, where other programs execute while the saving of a file was happening.
Scheduling is about choosing for how long a certain program can run before it needs to be interrupted for another program to be able to run. It is the operating system's job to choose for how long, and to choose the order the processes gets to run.

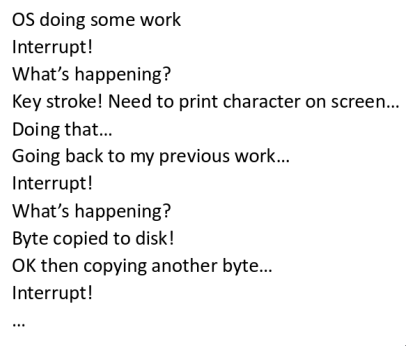

Figure 1.3: An example of a basic event cycle.

1.5 System calls (overview)

System calls are ways for user programs to *"talk to"* the hardware. All actions a user wants to do which modifies the hardware requires a system calls, e.g. Creating files, sending data over internet, showing something on a screen; are created through system calls. All system calls are types of interrupts. When the CPU finds the interrupt, control is given to the operating system, which chooses how to do the thing the user wants to do (if possible and the user should be able to).

1.6 Dual-Mode and Multimode Operation

What happens if a user tries to bypass the operating system by directly modifying the hardware without using system calls? The hardware keeps track of what kind of program is running, when the operating system is running, a bit is set which allows it to run "special functions" that normal users cannot.

To make sure the user cannot set that bit for themselves, when the hardware first starts, it loads the operating system into memory; during this phase the hardware and operating system defines which operations are privileged, and what code to run when those actions are performed. When the user tries to do those privileged actions, for example modifying a file, the operating system starts running.

1.7 Operating system services

1.8 System calls (continued) and APIs [Self reading]

1.9 System boot [Self reading]

1.10 Operating systems structures