Lecture 1: Introduction to Operating Systems

Barnaby Martin

barnaby.d.martin@dur.ac.uk

Overview

Lectures (9):

- Process Management
- Memory Management
- Storage Management
- File Management

Overview

Practicals (4):

- A: Command-line interpreter (next week)
- B: CPU scheduling algorithms
- C: Memory management
- D: Paging

(The Databases sub-module has practicals in the other weeks. Provisional timetable on DUO)

Operating Systems

Why do we need an operating system?

What is an operating system?

Operating Systems

Definition:

A program that acts as an intermediary between a user and the hardware.

Goals:

- Execute user programs.
- Make solving user problems easier.
- Make the computer system convenient to use.
- Use the resources of the system fairly and efficiently (priority, scheduling).

Operating Systems

An operating system is a:

- **Resource Allocator:** Responsible for the management of the computer system resources.
- **Control Program:** Controls the execution of user programs and operation of the input / output (I/O) devices.
- **Kernel:** The one program that runs all the time.

Processes

A process is a unit of execution; an abstraction that is used to support the discussion and study of operating systems.

Resources needed by the process include: CPU time, memory, files, and I/O devices.

The resources may be allocated at the start of a process or as it executes.

Executes (performs its work) using its associated resources; a collection of instructions that carry-out a reasonable task.

The operating system is responsible for process management including:

- Process creation and deletion
- Process holding and resuming
- Mechanisms for process synchronization (via priority, scheduling)

The Process Concept

A process includes:

- Code: text section
- Current activity, represented by the program counter and the content of the CPU's registers
- Data stack: Temporary data such as local variables
- Data section: Global variables
- Heap: Memory allocated while the process is running

Process Control Block

Information about each process is represented by a Process Control Block (PCB), including:

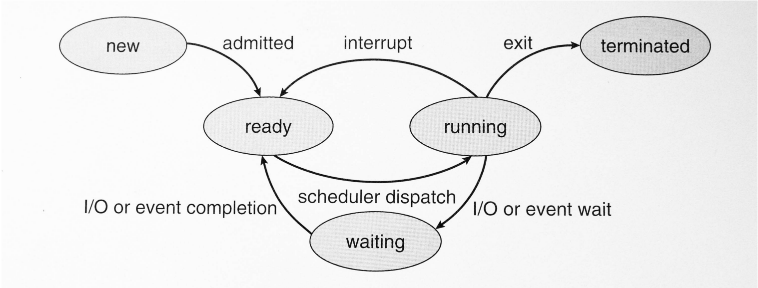
- Unique identifier e.g. PID
- State ($\sim$ status)
- CPU utilization
- CPU scheduling information e.g. priority
- Memory usage
- Other information: owner, description.

Process State

As a process executes, it changes state ($\sim$ status):

- **New** – the process is being created.
- Running – instructions are being executed.
- Waiting – the process is waiting for some event to occur.
- Ready – the process is ready to be dispatched to the CPU.
- **Terminated** – the process has completed its execution, or some other event causing termination.

Process State



Process Creation

A new process, as a parent process, can create a number of child processes, which, in turn create other processes, forming a tree of processes.

Resource sharing: three possible cases:

i. The parent and child processes share all resources.
ii. The child process shares a subset of parent's resources.
iii. The parent and child processes share no resources.

Execution: two possible cases:

i. The parent and child execute concurrently.
ii. The parent waits until child terminates.

Process Termination

The process executes its last statement and asks the operating system to delete it:

- Outputs data from the child's process to parent.
- The child process's resources are de-allocated by operating system

The parent process may terminate execution of child processes if:

- The child process has exceeded its allocated resources.
- The task assigned to child is no longer required.
- The parent is itself terminating (cascade termination).

The Kernel

Aim: To provide an environment in which processes can exist.

Four essential components:

- Privileged instruction set.
- Interrupt mechanism.
- Memory protection (see Memory Management lectures).
- Real-time clock.

The Kernel

The kernel consists of:

- The first-level interrupt handler (FLIH): to manage interrupts.
- The dispatcher: to switch the CPU between processes.
- Intra operating system communications e.g. via the system bus.

Interrupts

Definition

An **interrupt** is a signal from either hardware or software of an event that will cause a change of process, for example:

- **Hardware:** Triggers an interrupt by sending a signal to the CPU via the system bus e.g. I/O event (e.g. printing complete).
- **Software:** Triggers an interrupt by executing a system call for some action by the operating system (e.g. what time is it?).

Interrupt Routines: OS routines that execute whenever an interrupts occurs.

First-level interrupt handler (FLIH)

- The function of the FLIH is to:
  - Determine the source of the interrupt (prioritise).
  - Initiate servicing of the interrupt (selection of suitable process of the dispatcher).

Privileged instructions

Some instructions must be accessible only to the operating system: **privileged instruction set**.

Privileged instructions include functions such as:

- Managing interrupts (incl. enabling, disabling).
- Performing I/O.
- Halting a process!

Dual mode

Aim: To distinguish between execution of operating system code and user-defined code. We do not let the user execute instructions that could cause harm.

Two modes:

- User mode
- Kernel mode (or supervisor, system, privileged mode)

Privileged Instructions

Switching from user mode to kernel mode occurs when:

- a user process calls on the operating system to execute a function needing a privileged instruction (system call).
- an interrupt occurs (hardware).
- an error condition occurs in a user process (software).
- an attempt is made to execute a privileged instruction while in user mode.

The dispatcher

- Assigns processing resource for processes!
- Is later initiated when:
  - A current process cannot continue.
  - The CPU may be better used elsewhere, for instance:
    i. after an interrupt changes a process state.
    ii. after a system call which results in the current process not being able to continue.
    iii. after an error which causes a process to suspend.