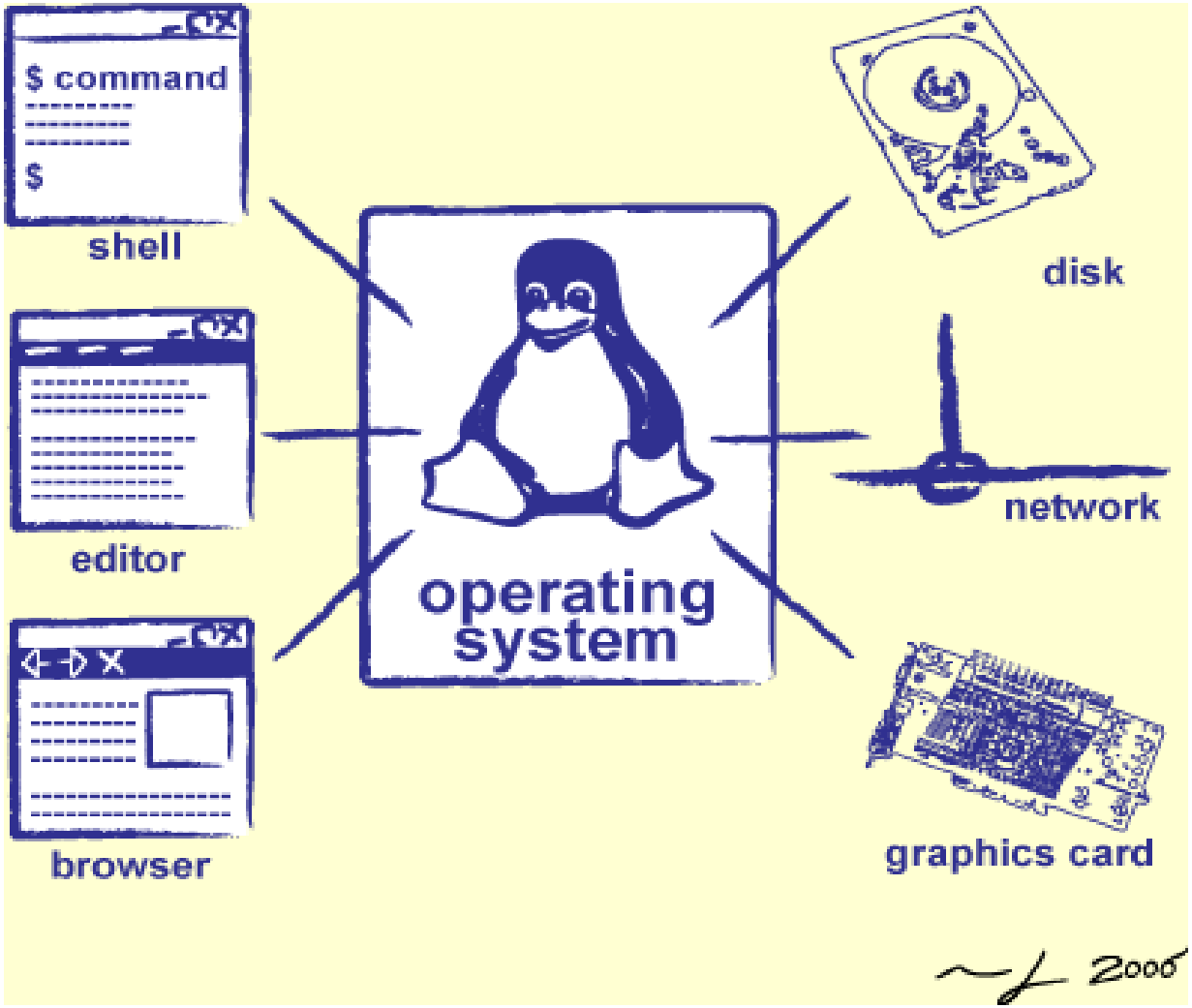
UNIVERSITY of STIRLING

www.cs.stir.ac.uk

Concurrent and Distributed Systems

Processes


$ command
$
shell
editor
browser
operating system
disk
network
graphics card
2006

Introduction

- About Processes
  - Execution of a program
  - Processes – an Operating System abstraction
  - Life cycle of a process
- Process Implementation
  - Parts of a process
  - Process Creation and Termination
- Cooperating Processes
  - Producer - Consumer Problem
  - Race Conditions

About Processes

- Informally: a program in execution
- Computer execute predefined actions
  - Actions are specified by a program
  - Program is a *selfcontained* entity
- Execution of a program requires resources
  - CPU
  - Running programs compete for CPU
  - Programs do not know when they get the CPU
- Actions of CPU and program logic are unrelated

About Processes

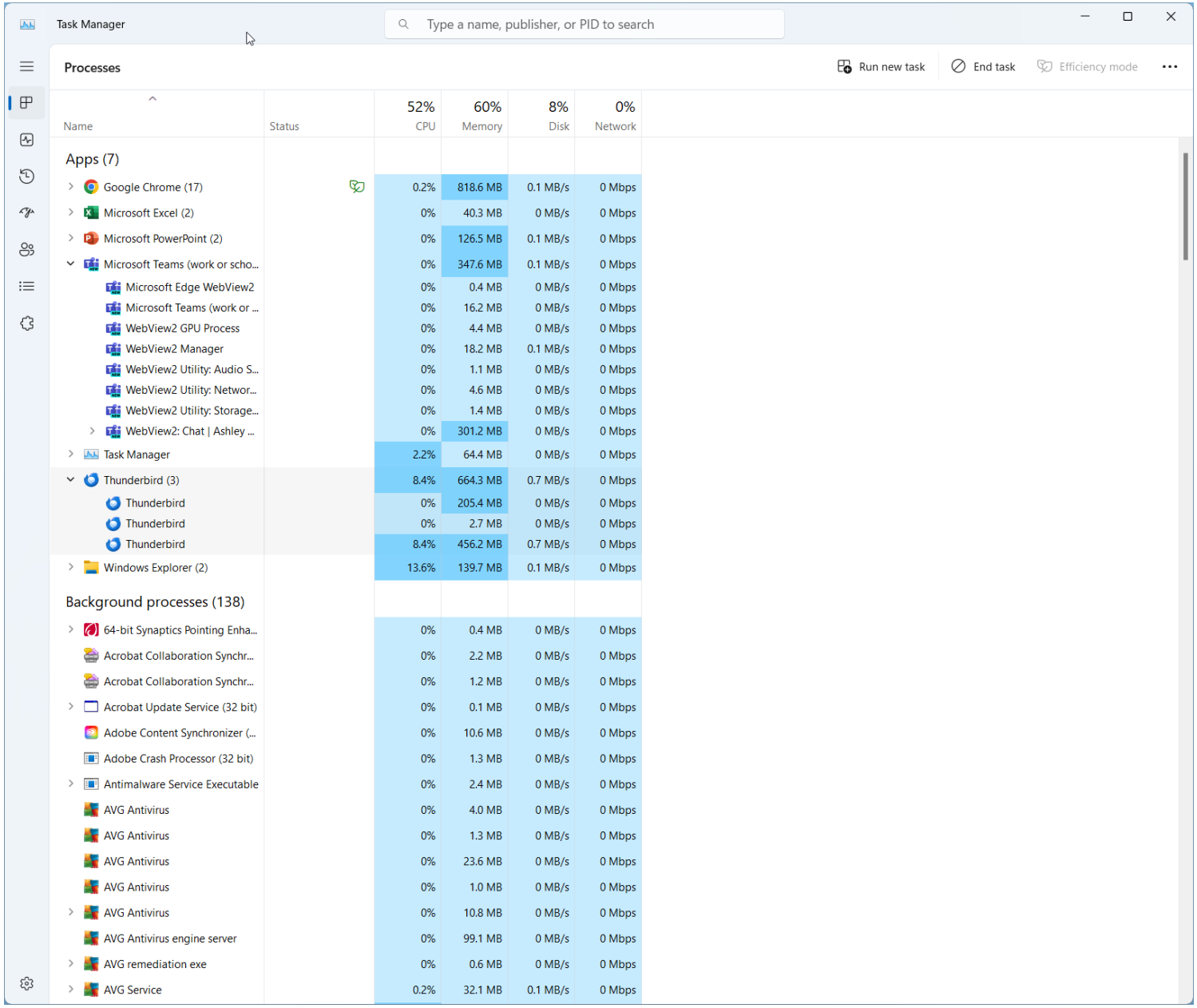
- Conclusions:
  - CPU is a normal reusable resource
  - For easy managing of running programs
    - Take the view of the program, NOT of the CPU
  - Abstraction of running programs is required!

- Program – Process relationship …

Program – Process Relationship

- A program is a static text that specifies a range of actions which have to be executed by one or more processors (passive entity).

- A Process is an entity executing the range of actions, which are specified by the program. Its thread of execution is somewhere in the middle between the first and the last of the actions specified by the program (active entity).


Task Manager
Type a name, publisher, or PID to search
Processes
Run new task
End task
Efficiency mode
Name
Status
52% CPU
60% Memory
8% Disk
0% Network
Apps (7)
Google Chrome (17) 0.2% 818.6 MB 0.1 MB/s 0 Mbps
Microsoft Excel (2) 0% 40.3 MB 0 MB/s 0 Mbps
Microsoft PowerPoint (2) 0% 126.5 MB 0.1 MB/s 0 Mbps
Microsoft Teams (work or scho... 0% 347.6 MB 0.1 MB/s 0 Mbps
Microsoft Edge WebView2 0% 0.4 MB 0 MB/s 0 Mbps
Microsoft Teams (work or ... 0% 16.2 MB 0 MB/s 0 Mbps
WebView2 GPU Process 0% 4.4 MB 0 MB/s 0 Mbps
WebView2 Manager 0% 18.2 MB 0.1 MB/s 0 Mbps
WebView2 Utility: Audio S... 0% 1.1 MB 0 MB/s 0 Mbps
WebView2 Utility: Networ... 0% 4.6 MB 0 MB/s 0 Mbps
WebView2 Utility: Storage... 0% 1.4 MB 0 MB/s 0 Mbps
WebView2: Chat | Ashley ... 0% 301.2 MB 0 MB/s 0 Mbps
Task Manager 2.2% 64.4 MB 0 MB/s 0 Mbps
Thunderbird (3) 8.4% 664.3 MB 0.7 MB/s 0 Mbps
Thunderbird 0% 205.4 MB 0 MB/s 0 Mbps
Thunderbird 0% 2.7 MB 0 MB/s 0 Mbps
Thunderbird 8.4% 456.2 MB 0.7 MB/s 0 Mbps
Windows Explorer (2) 13.6% 139.7 MB 0.1 MB/s 0 Mbps
Background processes (138)
64-bit Synaptics Pointing Enha... 0% 0.4 MB 0 MB/s 0 Mbps
Acrobat Collaboration Synchr... 0% 2.2 MB 0 MB/s 0 Mbps
Acrobat Collaboration Synchr... 0% 1.2 MB 0 MB/s 0 Mbps
Acrobat Update Service (32 bit) 0% 0.1 MB 0 MB/s 0 Mbps
Adobe Content Synchronizer (... 0% 10.6 MB 0 MB/s 0 Mbps
Adobe Crash Processor (32 bit) 0% 1.3 MB 0 MB/s 0 Mbps
Antimalware Service Executable 0% 2.4 MB 0 MB/s 0 Mbps
AVG Antivirus 0% 4.0 MB 0 MB/s 0 Mbps
AVG Antivirus 0% 1.3 MB 0 MB/s 0 Mbps
AVG Antivirus 0% 23.6 MB 0 MB/s 0 Mbps
AVG Antivirus 0% 1.0 MB 0 MB/s 0 Mbps
AVG Antivirus 0% 10.8 MB 0 MB/s 0 Mbps
AVG Antivirus engine server 0% 99.1 MB 0 MB/s 0 Mbps
AVG remediation exe 0% 0.6 MB 0 MB/s 0 Mbps
AVG Service 0.2% 32.1 MB 0.1 MB/s 0 Mbps

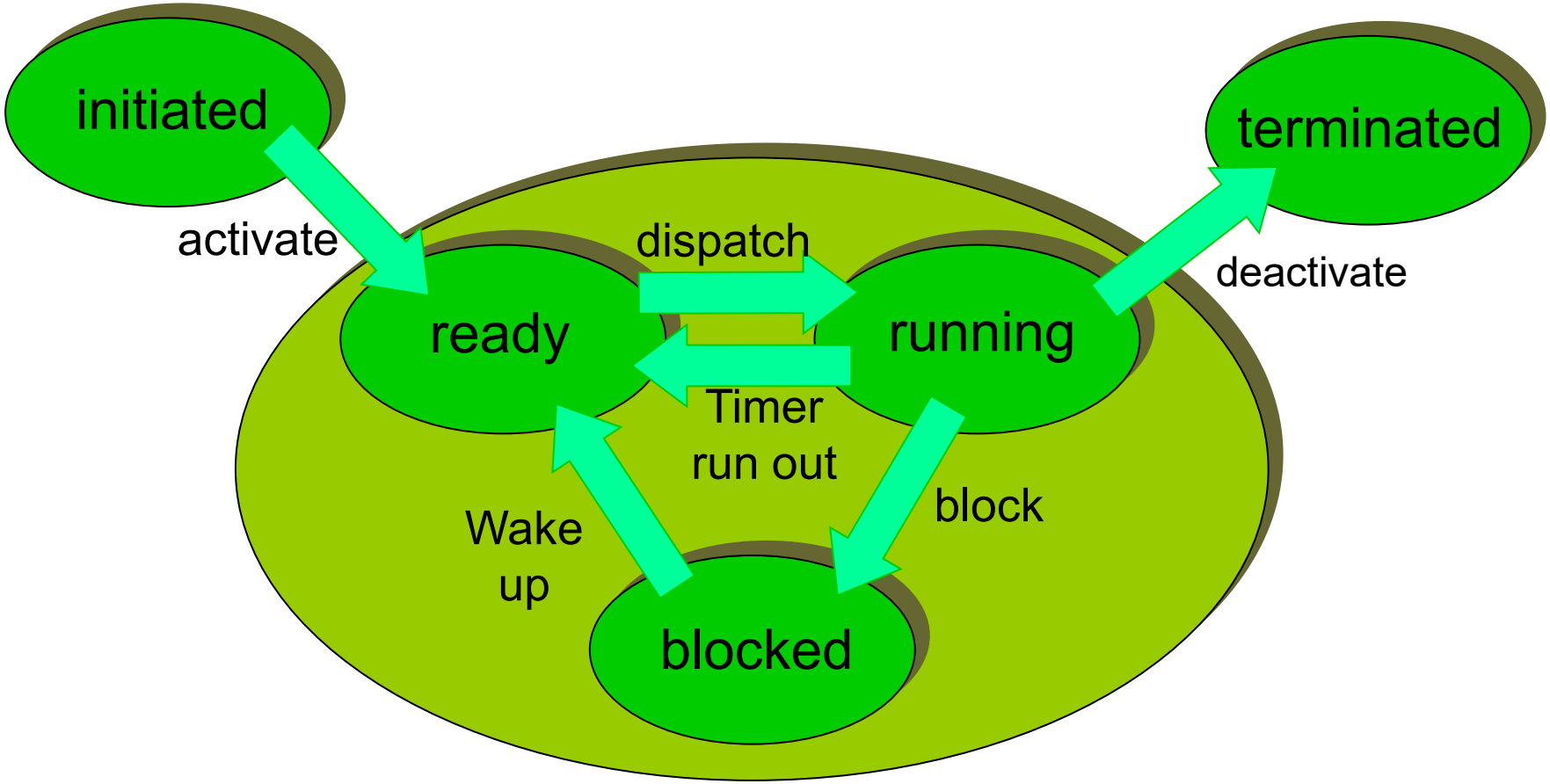
Processes Lifecycle

- Number of processes >> number of CPUs
- Some processes cannot run
- Processes waiting for resources
  - CPU (‘ready processes’)
  - Other resources (‘blocked processes’)
- Operating System provides for liveliness
  - State transitions of a process

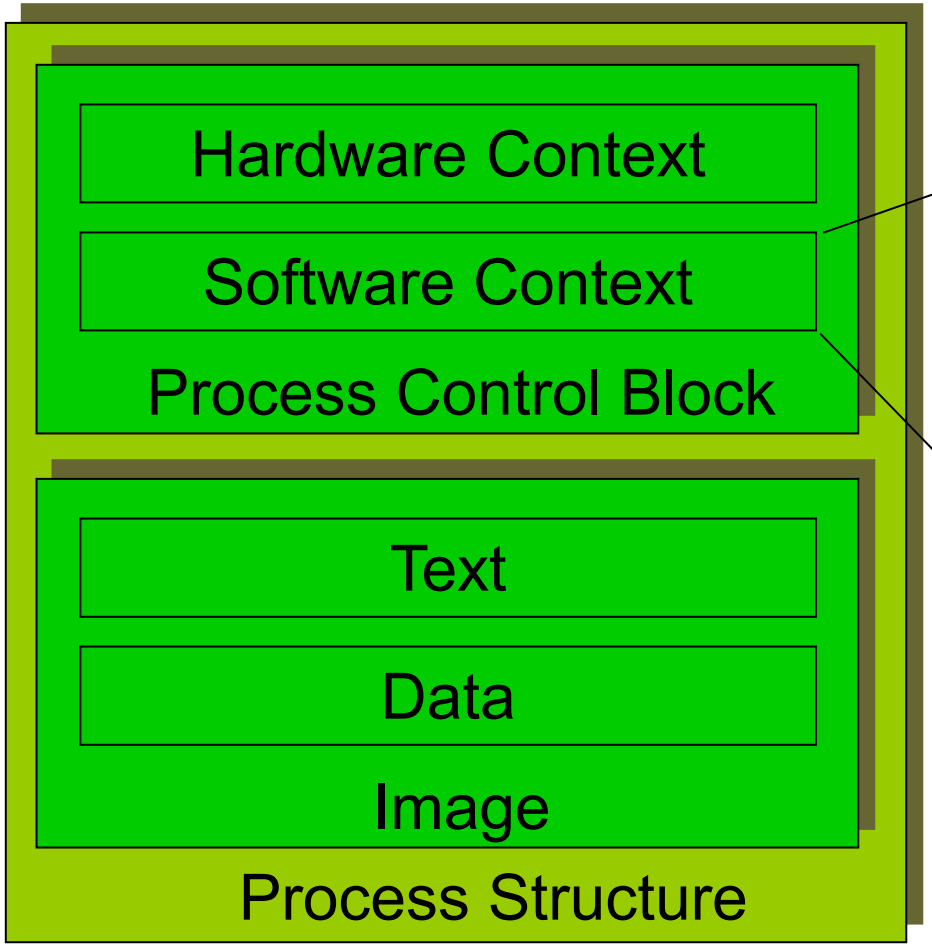
Process States

- As a process executes, it changes ***state***
  - initiated: The process is being created.
  - running: Instructions are being executed.
  - blocked: The process is waiting for an event to occur.
  - ready: The process is waiting to be assigned.
  - terminated: The process has finished execution.

Process Lifecycle

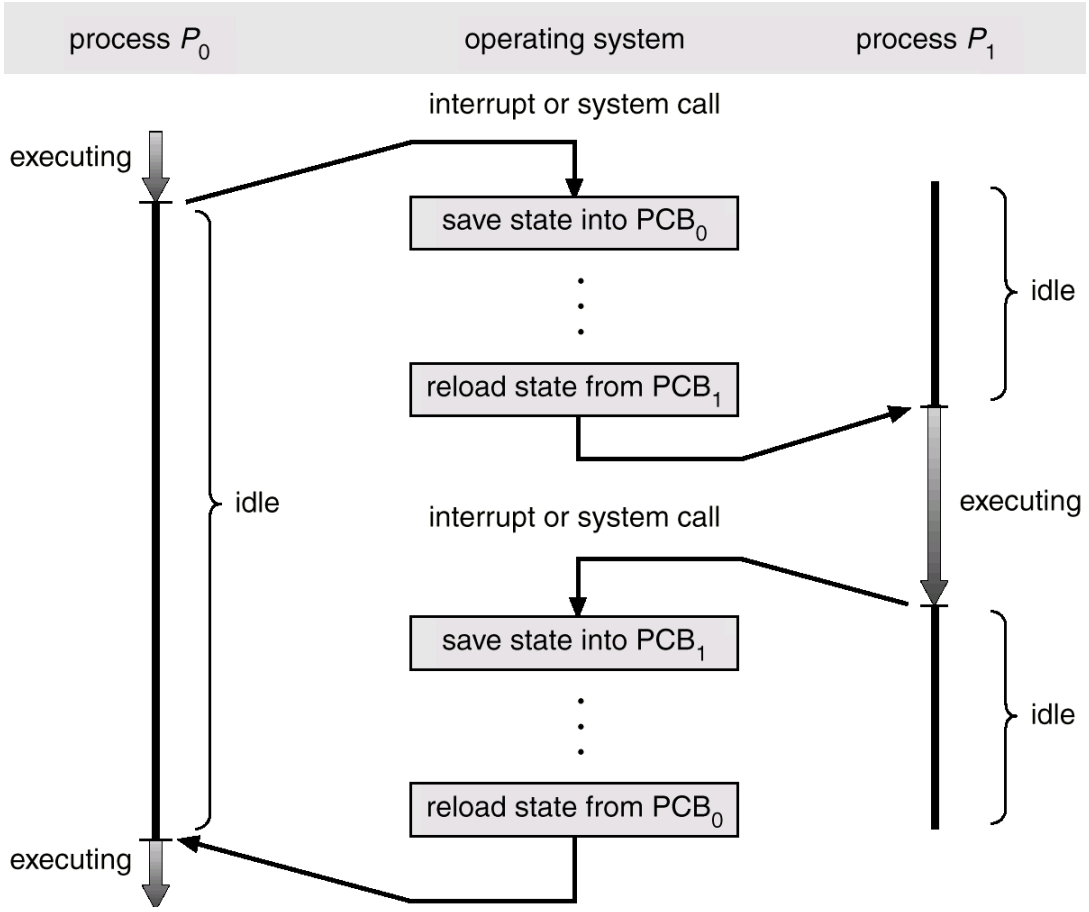


Parts of a Process



Process State
Program counter
CPU scheduling information
Memory management inform.
Accounting information
I/O status information

Context Switch



- When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process.
- Context-switch time is overhead; the system does no useful work while switching.
- Time dependent on hardware support.

Process Creation

- Parent process creates children processes, which, in turn create other processes, forming a tree of processes.
- Resource sharing
  - Parent and children share all resources.
  - Children share subset of parent's resources.
  - Parent and child share no resources.
- Execution
  - Parent and children execute concurrently.
  - Parent waits until children terminate.
- Address space
  - Child duplicate of parent.
  - Child has a program loaded into it.

Example

```
#include <stdio.h>

void main(int argc, char *argv[])
    {
    int pid;

    /* fork another process */
    pid = fork();

    if (pid < 0) { /* error occurred */
            fprintf(stderr, "Fork
                        Failed\n");
            exit(-1);
    }
    else if (pid == 0) { /* child process */
            execlp("/bin/ls","ls",NULL);
    }
    else { /* parent process */
     /* parent waits for the child to
                                    complete */
            wait(NULL);
            printf("Child Complete\n");
            exit(0);
    }
}
```

Cooperating Processes

- *Independent* process cannot affect or be affected by the execution of another process.
- *Cooperating* process can affect or be affected by the execution of another process
- Advantages of process cooperation
  - Information` sharing (shared files)
  - Computation speed-up (split-up of a task into subtasks and run them in parallel; Note: number of processors!)
  - Modularity (divide a system into separate processes)
  - Convenience (a user has many tasks; e.g. printing, editing, compiling)

Producer-Consumer Problem

- Paradigm for cooperating processes, *producer* produces information that is consumed by a *consumer*.
  - Examples: printer queue, keyboard buffer
  - Consumer and Producer processes need to be synchronised
  - Buffer may be provided by OS (IPC mechanism) or be explicitly coded by the programmer using main memory, file, etc.
  - *unbounded-buffer* places no practical limit on the buffer size
    - Producer can always produce elements
    - Consumer cannot retrieve an element when there are no elements
  - *bounded-buffer* assumes that there is a fixed buffer size.
    - Producer cannot add an element when the buffer is full
    - Consumer cannot retrieve an element when there are no elements

Race Conditions

- Concurrent access to shared data may result in data inconsistency.
- Maintaining data consistency requires mechanisms to ensure the orderly execution of cooperating processes.
- Example: Print Spooler (try and spot the similarity to the first practical)

Example of Race Condition

- Print Spooler
  - User processes put file names into a spooler directory
  - Printer daemon periodically checks for any new files in the spooler directory
    - If there are any – print the file and remove file name from spooler directory
  - Directory has infinite number of slots
  - Two globally available variables:
    - *out* points at the next file to be printed
    - *in* points at the next available slot
  - At a time: slots 0 – 3 are empty (files printed)
    slots 4 – 6 are full (files to be printed)

Example cont.

- Almost simultaneously processes A and B want to print
- Process A reads *in* (value 7) and assigns it to local variable
- Process context switch occurs to process B
- Process B reads *in* (value 7) and stores a file in slot 7, updates *in* to 8
- Process A runs again, continuing from where it stopped
- Process A reads local variable and stores a file at slot 7
- Process B's file is erased!
- RACE CONDITION!