Comp 204: Computer Systems and Their Implementation

Lecture 2: Processes

Today

- OS evolution
- Introduction to processes
- OS structure

Evolution of OS

- Largely driven by desire to do something useful when a program cannot continue (maximise throughput)

- Early systems:
  - 'Job' loaded from punched cards or tape, output to printer
  - Job may include loading compiler, assembler, linker, data etc.
  - CPU idle for much of the time

- Batch systems:
  - Job passed to human operator
  - Operator groups jobs into batches with similar characteristics, e.g. all programs using same compiler
  - More efficient use of resources

Multiprogramming

- Load several programs into memory simultaneously, all sharing single CPU
- When running program cannot continue (e.g. waiting for I/O), switch to another
- Hence, I/O and computation overlap

Multi-Access (Time-Sharing)

- An extension of multiprogramming

- CPU is switched rapidly between processes to give illusion of uninterrupted execution in parallel (multitasking)
  - users can interact with programs
  - users see their own ‘virtual machine’
  - resources (printers, disks etc.) are shared, but this is largely transparent

Question

- The following two statements describe the performance of two programs (where the computation and input/output could be interleaved):
  - A performs a total of 20 seconds of computation and 15 seconds of input/output.
  - B performs a total of 30 seconds of computation and 10 seconds of I/O
- Which of the following are true?

I. It will take up to 50 seconds to run A and B sequentially
II. It will take up to 75 seconds to run A and B sequentially
III. Using multiprogramming, the shortest time to execute both is 50 seconds
IV. Using multiprogramming, the shortest time to execute both is 40 seconds

a) I and III
b) I and IV
c) II and III
d) II and IV
e) None of the above

Answer: c

- If run sequentially, A needs to finish before B can begin, therefore II is true.
- With multiprogramming, I/O for one process can take pave whilst the computation takes place for another. Therefore III is true

Implications

- Need to decide which programs to load from disk into memory (job scheduling)
- Need to decide which program to execute next (CPU scheduling)
- Consider disk space as extension of main memory (virtual memory)
- Memory allocation
- Disk/file allocation
- Protection/security

Personal Computers

- Originally intended for single users
- Development concentrated on usability (GUIs etc.)
- Now incorporate many features from larger systems
  - multitasking
  - networking
  - printer and file sharing
  - security

Example – IBM PC

- Single tasking: MS-DOS
  - To run program, command interpreter over-writes part of itself with program, then transfers control
  - When program completes, execution returns to OS, which then reloads rest of interpreter
  - Limited concurrent execution possible via TSR (terminate and stay resident) system call
- Multiprogramming, non-preemptive: Windows 3.x
- Full multi-tasking: Windows 95 onwards, Linux

Parallel Systems

- Most systems are single–processor (uniprocessor) systems: they have one main CPU
- However, there are systems that have more than one processor that communicate and share resources.
- These are known as multi-processor systems
- Purpose:
  - increasing the number of processors should enable more work to be done in less time (maximising throughput)
  - Reduces costs when resources are shared
  - Increases reliability: the failure of one processor will not halt the system (though it will slow it down)

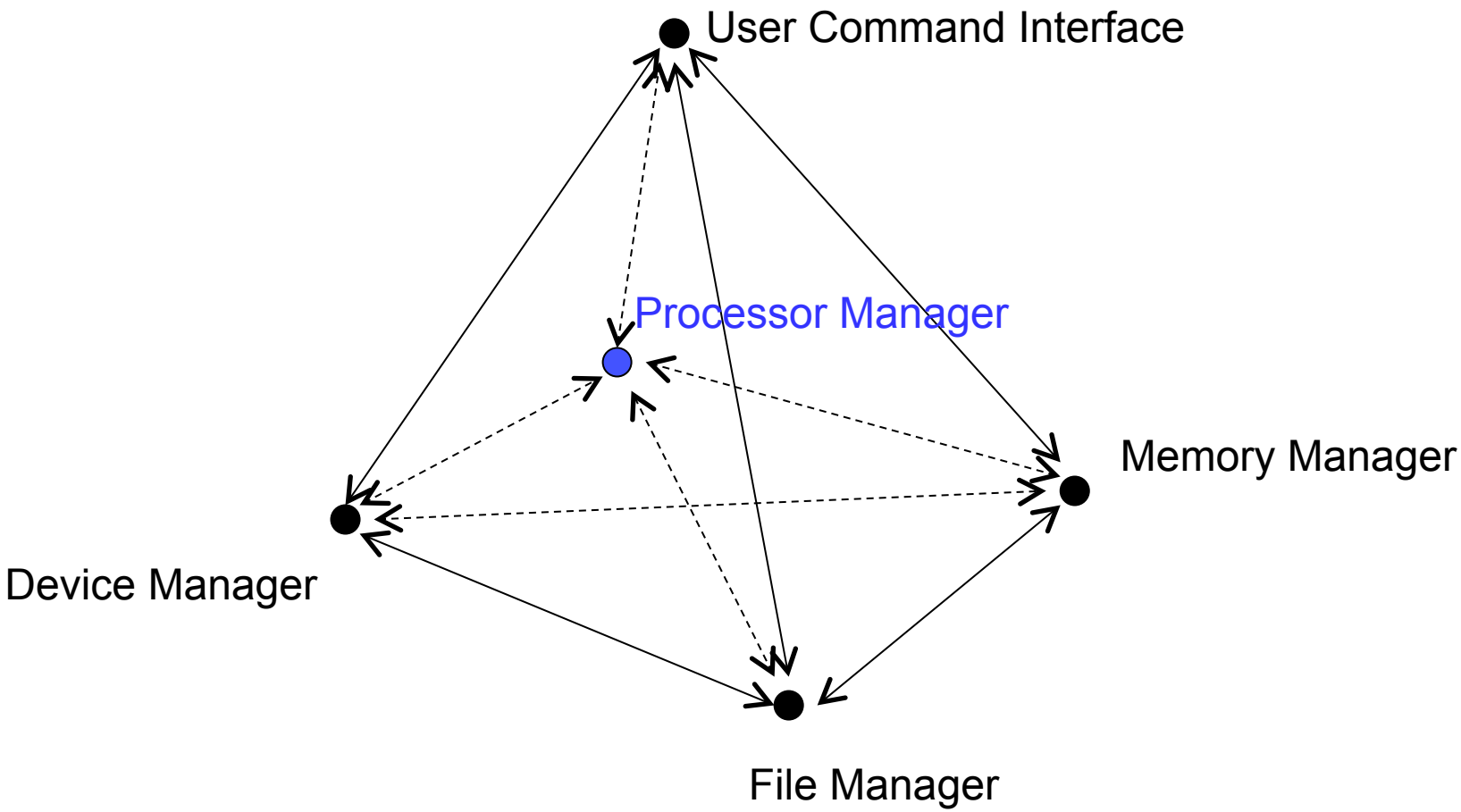
Real-Time Systems

- Real-time systems: special purpose OS used when there are strict time constraints on the operation of a processor or the flow of data
- Often used as a control device in a dedicated application
- Requires delays in the system to be bounded
  - Time constraints on retrieval of stored data
  - Time constraints on how long it takes the OS to finish any request made of it
- Some facilities are absent from such systems:
  - Secondary storage limited or absent
  - Advanced OS features separating user from hardware absent, e.g. virtual memory
- Examples of application areas: multimedia, virtual reality, scientific projects

Distributed Systems

- Distributed systems: relatively recent development due to growth of networked systems, esp. WWW: PCs can access WWW through browsers
- Many current OS include system software to enable a computer to access the Internet via a local-area network (LAN)
- Such systems provide network connectivity, though some OS take the concept further:
- A network OS is one that stands alone from the other computers on the network but can communicate with the other networked computers
  - Provides features such as file sharing and communication across the network
- There are also distributed OS that operate less autonomously: the different OS communicate closely enough to create illusion of a single OS controlling the network

Operating System – An Abstract View



Processes

- A *program* is a representation of an algorithm in some programming language; i.e. it is *static*
- A *process* refers to the activity performed by a computer when executing a program; i.e. it is *dynamic*
- A process is created when a program or command is executed

Question

- Suppose two users simultaneously type the following command at the unix shell command prompt ($):

```
$ ls -l
```

- Which of the following are true?

a) One process and one program is involved
b) Two processes and two programs are involved
c) One process and two programs are involved
d) Two processes and one program are involved
e) None of the above

Answer: d
Only one program (ls) is involved, but this will be run as two pocesses.

Process Characteristics

- Process characteristics:
  - Requires space in memory where it resides during execution
  - During its execution it may require other resources such as data files or I/O
  - It passes through several states from its initial creation to its completion within the computer system (more details on these states to come in later lectures)

Processes

- A process needs resources, such as CPU time, memory, files and I/O devices, to accomplish its task.
- These resources are allocated either when the program is created, or when it is executing.
- Operating-system processes execute system code and user-processes execute user code
  - All these processes could potentially execute concurrently

Processes

- The Processor Manager is responsible for overseeing the following activities in relation to process management:
  - Creation and deletion of both system and user processes
  - Scheduling processes
  - Provision of mechanisms for synchronisation and communication of processes
  - Deadlock handling for processes

O.S. Structure

- Often consists of:
  - A central nucleus or kernel
    - resides permanently in memory
    - performs low-level, frequently needed activity
  - A set of processes
    - may be system level or user level
  - processes interact with kernel via system calls
    - e.g. create process, run program, open file
  - kernel and system level processes may operate in privileged mode

Command Interpreter

- Accepts and runs commands specified by user
  - Hence provides user's view of OS
- May be graphical, e.g. Windows
- May be textual, e.g. UNIX shell
  - bash, ksh, csh
  - Some commands built into shell, others loaded from separate executable files
  - Shell also has sophisticated control structures such as loops, if-statements and procedures