From last time

An OS may contain managers for Devices, Network, Filestore, Memory, & Processes. Which would be in an OS for:
– A process control computer with a sensor for monitoring, an actuator for control, and a network connection for reporting to and receiving commands from a control centre?
– A dedicated, network-based filing machine or ”file server”?

– A computer dedicated to controlling the communications passing between two networks; that is, a ”gateway”?

– An autonomous lap-top personal computer?
– A single-user workstation with services available across a network?
– A machine dedicated to managing and answering queries on a database?

MANCHESTER 1824

The University of Manchester

COMP25111: Operating Systems

Lecture 4: Operating System Concepts

John Gurd

School of Computer Science, University of Manchester

Autumn 2014

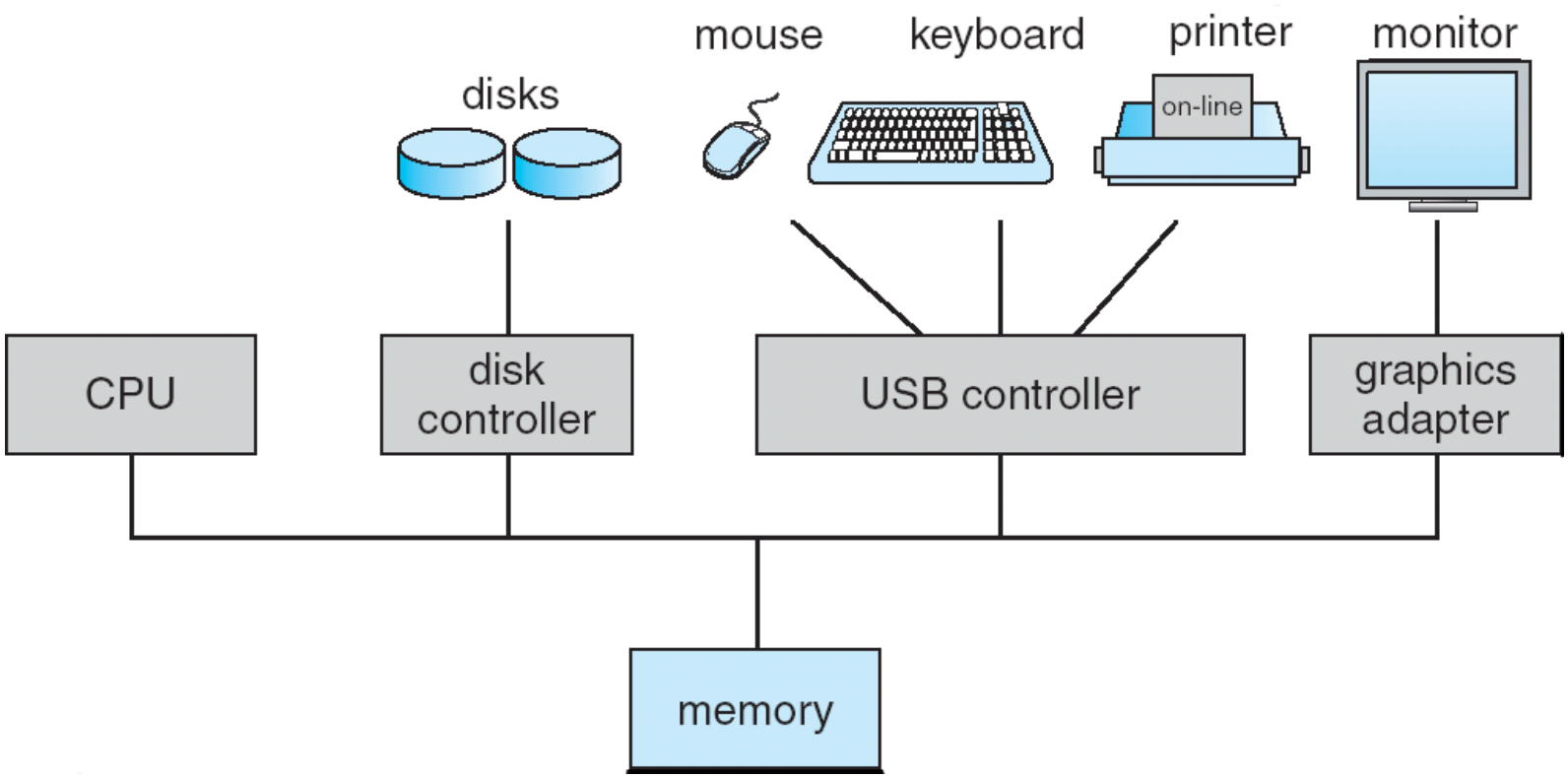
Overview & Learning Outcomes

Overview of (multi-programming) OS
– functions & components

Processes

Protection

Components of a simple PC



– details of devices are hidden from Apps
– several things can be happening at once

What does an Operating System Do?

Manage Resources:
– multiple devices $\rightarrow$ deal with concurrency
– sharing
– protection

Provide services:
– multiple Apps $\rightarrow$ provide concurrency
– abstraction
e.g. filestore, not disk drive
e.g. variable size stack
e.g. reliable network connection

Process = Thread + Address Space

Process: a program in execution (not a program on the disk)

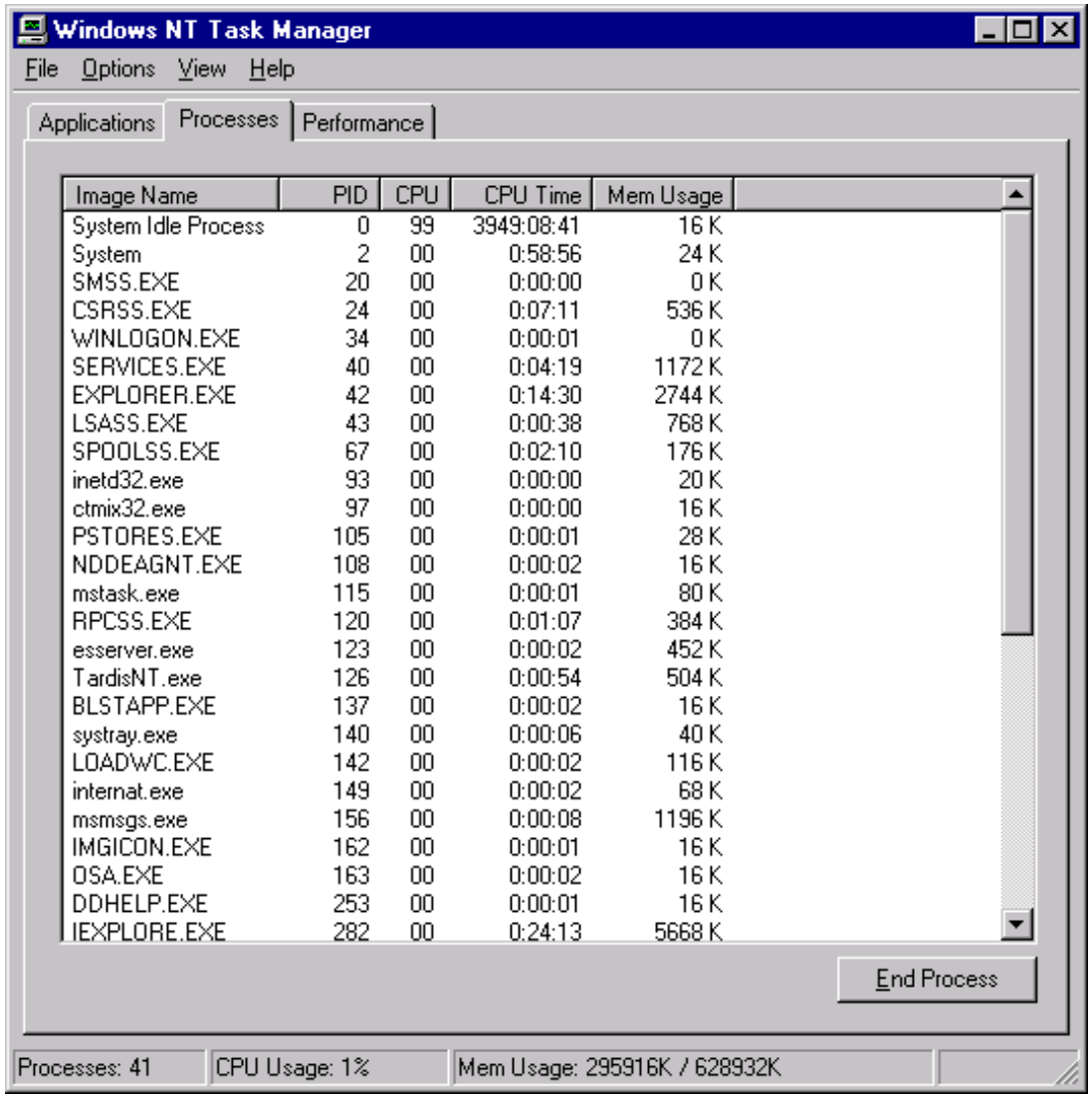
Address Space: all memory locations the process can use

Thread: "of execution" – sequence of instructions obeyed

Multi-threading: multiple threads within the same process

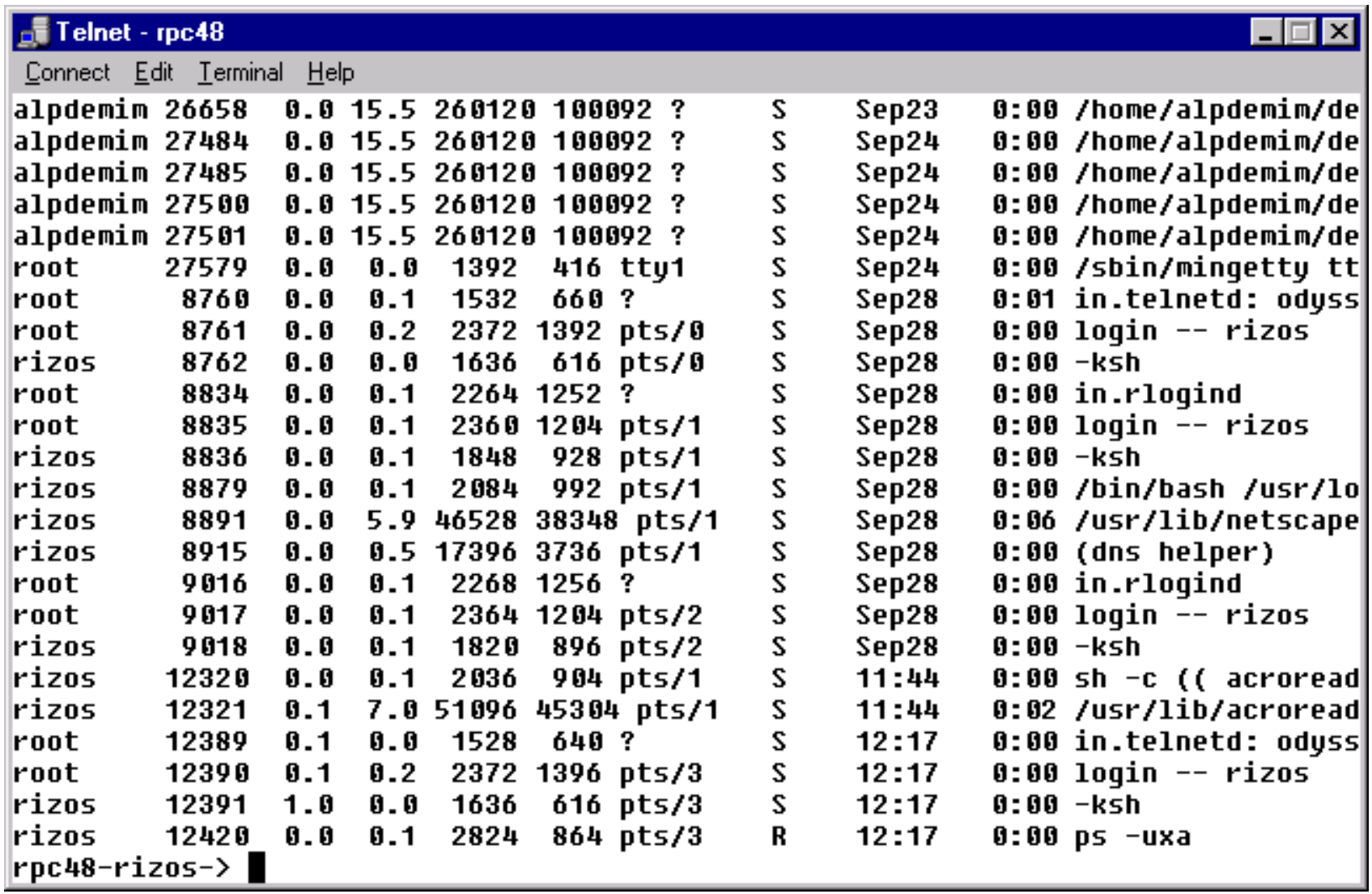
Many processes exist at any time

Windows XP: <CTRL><ALT><DEL>



Many processes ctd.

Linux: `ps uxa`



Address Spaces

e.g. ARM/MU0 assembler addresses start at 0

But, several programs can be in memory at the same time - each assuming this

OS may pause a running program, swap it out of memory & later swap it back to somewhere different

Relocation - how to make each program think it has sole use of memory

Relocation example: a C program

```
int x;
main (int argc, char *argv[]) {
  x= atoi(argv[1]);
  printf("%d %p\n", x, &x);
}
```

e.g. `./a.out 7` from two different Linux shells
both output: `7 0x8049678`

Different programs seem to use the same address

Virtual Machine

OS provides “Virtual Machine”
– more convenient abstraction than real machine
– Apps think they use the hardware on their own

Virtual Machine enforces Protection:
– System v. Program
– Program v. Program

OS needs hardware support – execution mode:
– User mode
– System (Privileged, Supervisor) mode

Privileged Operations

OS components run in System mode

OS runs Apps in User mode

H/W prevents certain operations in User mode:

– memory operations?

– CPU allocation?

– I/O operations?

– file operations?

– network operations?

System call

How do Apps use protected resources?

System call: interface between Apps & OS

like method/function call – parameters, caller waits for result

via “gatekeeper” mechanism (H/W + OS)
– turns on System mode
– calls OS routine from list
– parameters etc. checked
– action performed
– returns to User mode

Details vary between OSs, underlying concepts similar

System Call example

Unix “read” has 3 parameters: the file, where to put the data, how many bytes to read

```
read(int fd, char *buf, int num_bytes);
```

Not the **C library function**:

```
fread(void *ptr, size_t size, size_t n, FILE
*stream);
```

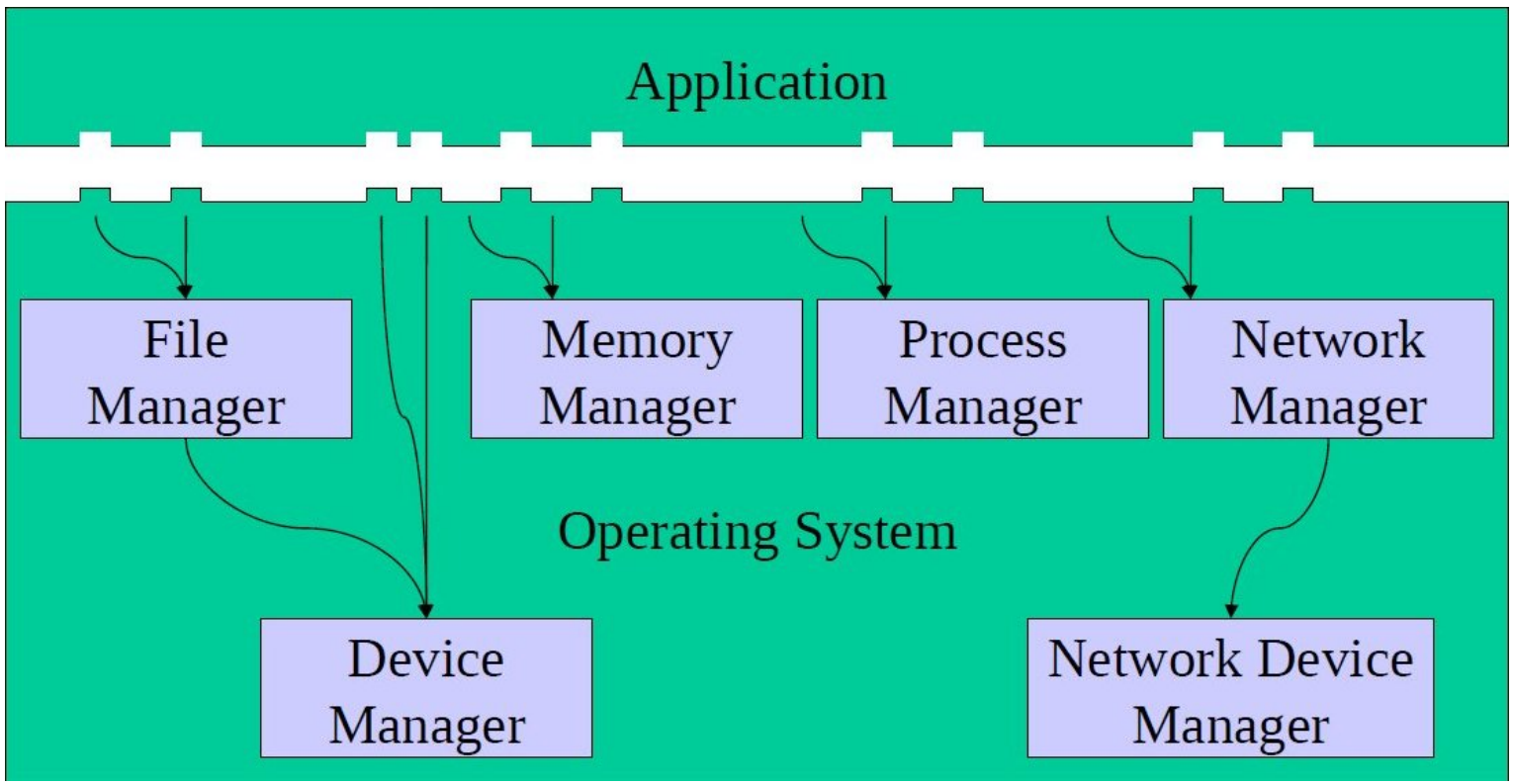
– library functions can do more
– not all library functions correspond to system calls

Many languages do not allow system calls to be made directly

OS Components

A system so large and complex can be created by partitioning into smaller pieces

Most OSs have different structures



OS Components provide services

Process Management: creation, deletion, CPU allocation, ...

Memory Management: Allocate and deallocate memory space;
Keep track of what parts of the memory are being used, ...

Device (I/O) Management: read & write bytes

File (and Secondary Storage) Management: ...

Network Management: ...

User interface: GUI, command line interpreter (shell)

User/App use services

e.g. User types `run myprog` (just `myprog` in Unix)

– read command (command interpreter/shell)
– find program file (how big?)
– allocate memory
– read file into memory
– find libraries
– start myprog running
– finish “cleanly”

Also: accounting, security, error detection/reporting, ...

Engineering an OS...

Monolithic systems (no structure - the “big mess”)

Layered approach (bottom = H/W, highest = U.I)
Layers selected so each only uses functions, operations & services of lower layers.
Lower layers (“**kernel**”) contain most fundamental functions to manage resources.

Big OS Kernels have problems (complexity, debugging)
several Mbytes (linux 2-3)

Microkernels keep only minimal functionality in the OS

Summary of key points

Process = Thread + Address Space

Protection: Virtual Machine
– H/W support: User mode v. System mode
– System calls for Priviledged operations

OS Structure
– Components (Managers): Process, Memory, I/O, File, ...
– Layered, Kernel, Micro-Kernel

Next time: Process Management

Your Questions

For next time

Which of the following operations would you expect to be privileged (available only in System mode) & which available in User mode?

– halt the processor?
– system call?
– write an absolute memory location?
– load register from memory?
– disable interrupts?
– load stack pointer?
– write to segment or page not present in memory?
– change memory management register value?
– write to Program Status Register?
– write to interrupt vector table?

Exam Questions

Why do computers typically have two modes of operation, namely user mode and system mode (also known as supervisor or kernel or privileged mode)? (2 marks)

Explain briefly what is a system call (2 marks)

What does it mean to say that a system is constructed using the ”micro-kernel approach”? (2 marks)

Glossary

- Device
- Resource
- Concurrency
- Process
- Address space
- Thread
- Multi-threading
- Relocation
- Virtual Machine
- System/Supervisor/Priviledged mode
- User mode
- System call
- Library function
- Manager
- Monolithic OS
- Layered OS
- OS Kernel
- Microkernel OS

Reading

OSC/J: Chapters 1 & 2

MOS: Sections 1.5-1.11 (skim through the system call details)

(both books use some concepts in these sections that will be clarified later on)