ELEC 377 – Operating Systems

Matthew Stephan

Week 1 – Class 1

Instructor

Matthew Stephan

Office:	Goodwin Hall 624
Email:	matthew.stephan@queensu.ca
Hours:	Wed 1:30
	and by appointment

- Engineering Undergrad and Masters at UWaterloo
- Ph.D. Candidate in CISC
- 2 years industrial experience
- 2 years teaching experience

ECE Rep

Election/Volunteer

Required Textbooks

Operating System Concepts, 8th Ed

Silberschatz, Galvin, Gagne

7th edition is acceptable.

Some Sort of C Reference (not C++) is *required*

- You can use what you feel comfortable with

Notes

- I may be away for the 3rd week of class for a conference
    - I will assign a guest lecturer.
- You are responsible for what is in the text, what is on the slides and what is said in class.
    - If you miss a class, make sure you get notes from someone that was there
    - There will be material not in the textbook

Marking Scheme

- 5 Assignments done in C/bash on Linux
  (25% of total mark)

- 3 Quizzes
  (8% each, 24% of total mark)

- Final Exam (51% of total mark)

Assignment Marking Scheme

- Questions + Programming

Programming Part:

- Documentation (2 parts) 40%
  Description of the problem and your solution

- Structure and Clarity of Approach 20%
  Code must be well structured, clear and commented.

- Testing and Correctness 40%
  Test cases and output must be included in what is handed in. Must also include a written argument about the completeness of your testing

Quizzes

- There will be 3 quizzes, each 20 minutes long
- Quizzes will be given in class at the beginning of the lecture
  - Each quiz will cover the lecture material from the previous quiz up to and including the immediately preceding class. You are responsible for all material covered in class. Some subjects have extra material in class not in the textbook.
- The tentative quiz dates are Tuesdays (Week 3,6 and 9)
  Sept 25 Oct 16 Nov 6

Tutorial and Labs

- Tutorial and Labs are combined in the Teaching Studio (ILC 213)
- One Section, 10:30 Monday (2hrs)
  - Teams of two
  - Please choose a partner by next Monday and send me email with your partners name.
- Most labs are multiple weeks long
  - important to be prepared
- The first lab is an Orientation Lab, we will go through a simple exercise of modifying the Linux kernel. Even though there are no marks, it is important you be there to make sure that your environment works.

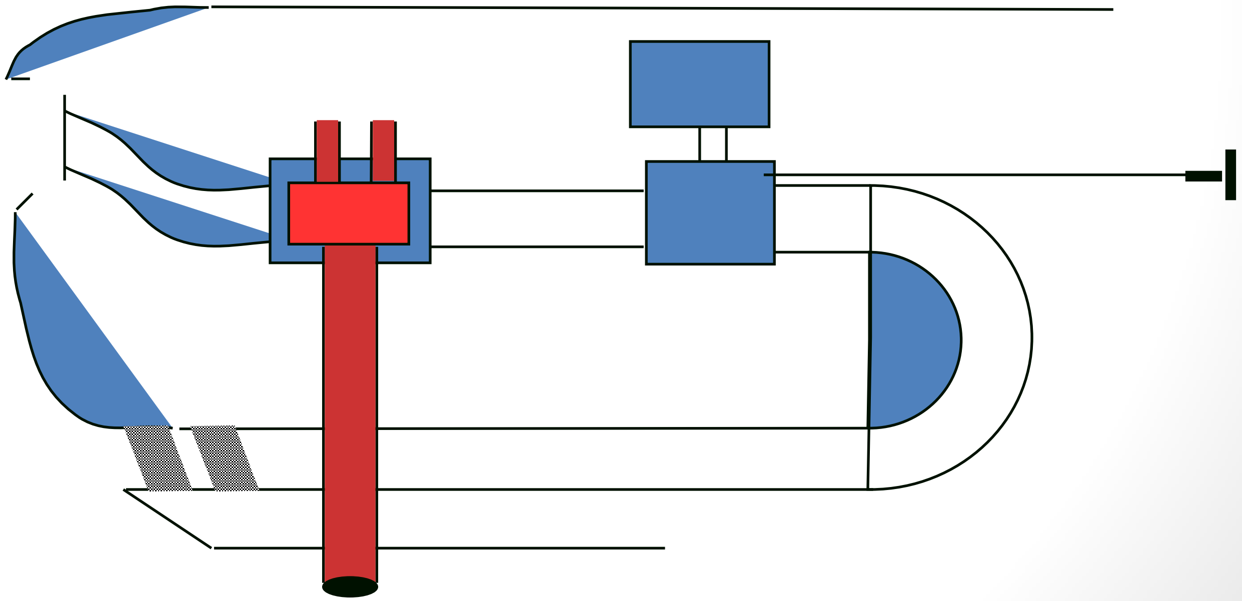
Tutorial and Labs

- There will be an extra tutorial this week. This will be an in–depth tutorial of C programming for the course. Attendance is optional. However if you have no C programming experience beyond ELEC 276 you **should be** there.

Monday September 10th at WLH 210 at 6:30PM

Why Study Operating Systems?

- Sophisticated Users
  - When programming, operating issues become relevant. Threads, priority, sharing.
  - Car Driver vs. Airplane Pilot Model



Adapted from: Transport Canada Flight Training Manual 3rd ed.

Why Study Operating Systems?

- Interaction between Hardware and OS

– Modern hardware contains many features needed to support operating system functionality

– Memory/Process/IO Protection

– Interrupt, Cache, and Virtual Memory features

– Hardware designers need to understand OS

Why Study Operating Systems?

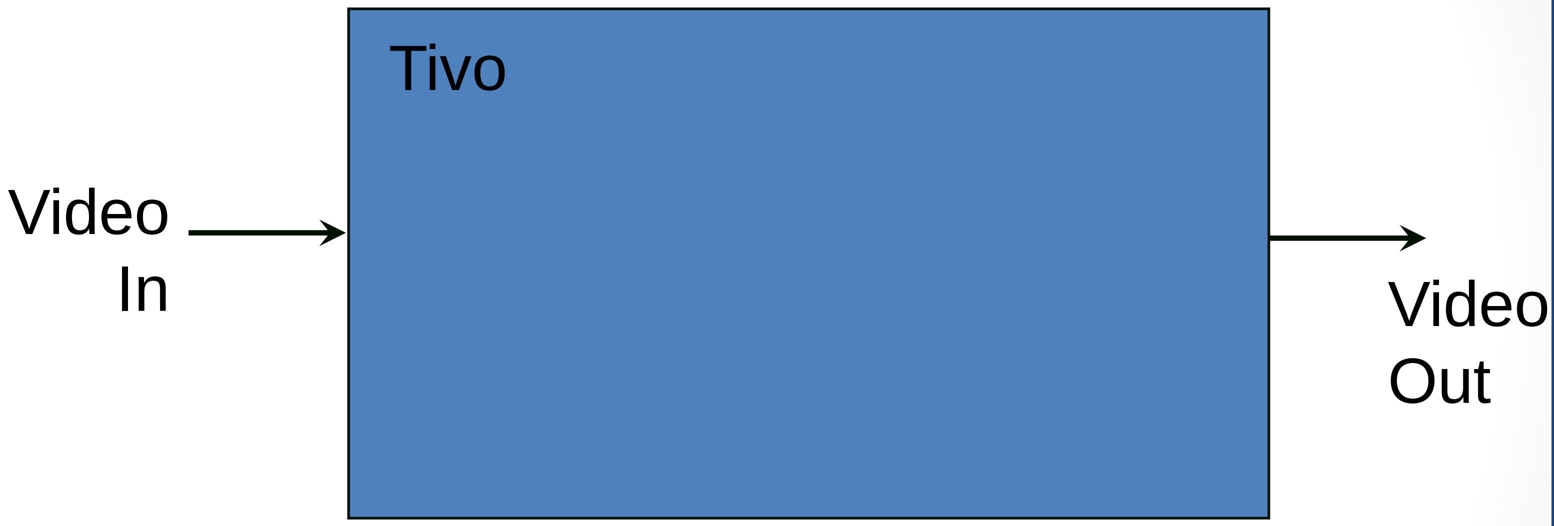
- Modern Embedded Systems
  - There is a continuum between small devices and more sophisticated devices
  - For small devices such as Interact Terminals, minimal OS support is needed (ELEC 371)
  - Fly-by-wire systems need more support
  - Growth in consumer devices (TiVo, LG Fridge)

- Informed User
  - A lot of misinformation out there from all parties.
  - You may have to choose and recommend for employers and clients

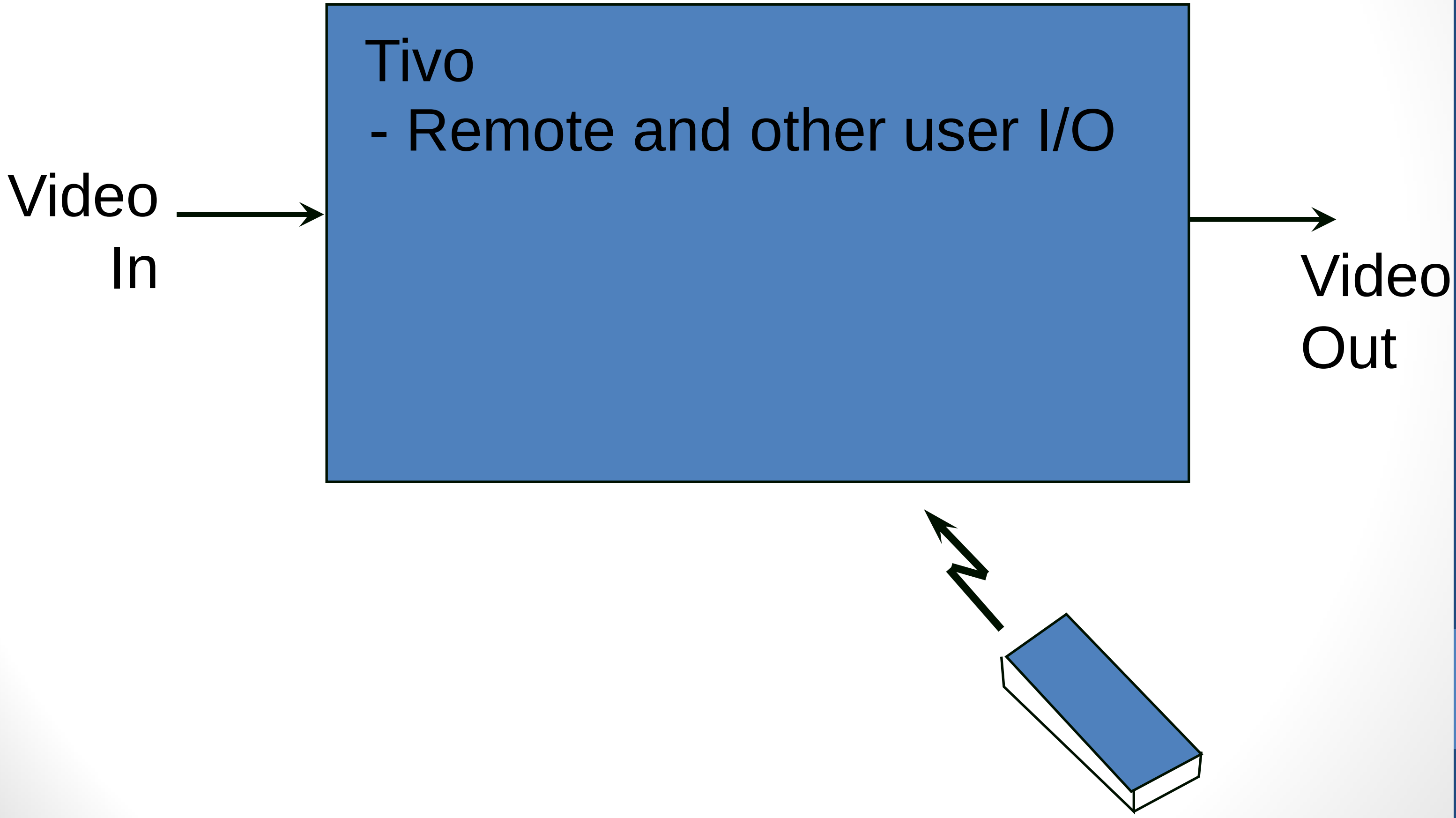
TiVo – What is it?

- Digital Video Player
  - Like VCR, but no tapes (hard drive)
  - Customized Guide (record program by name)
  - Phone network connection for correct time and content guide
  - Pause live events

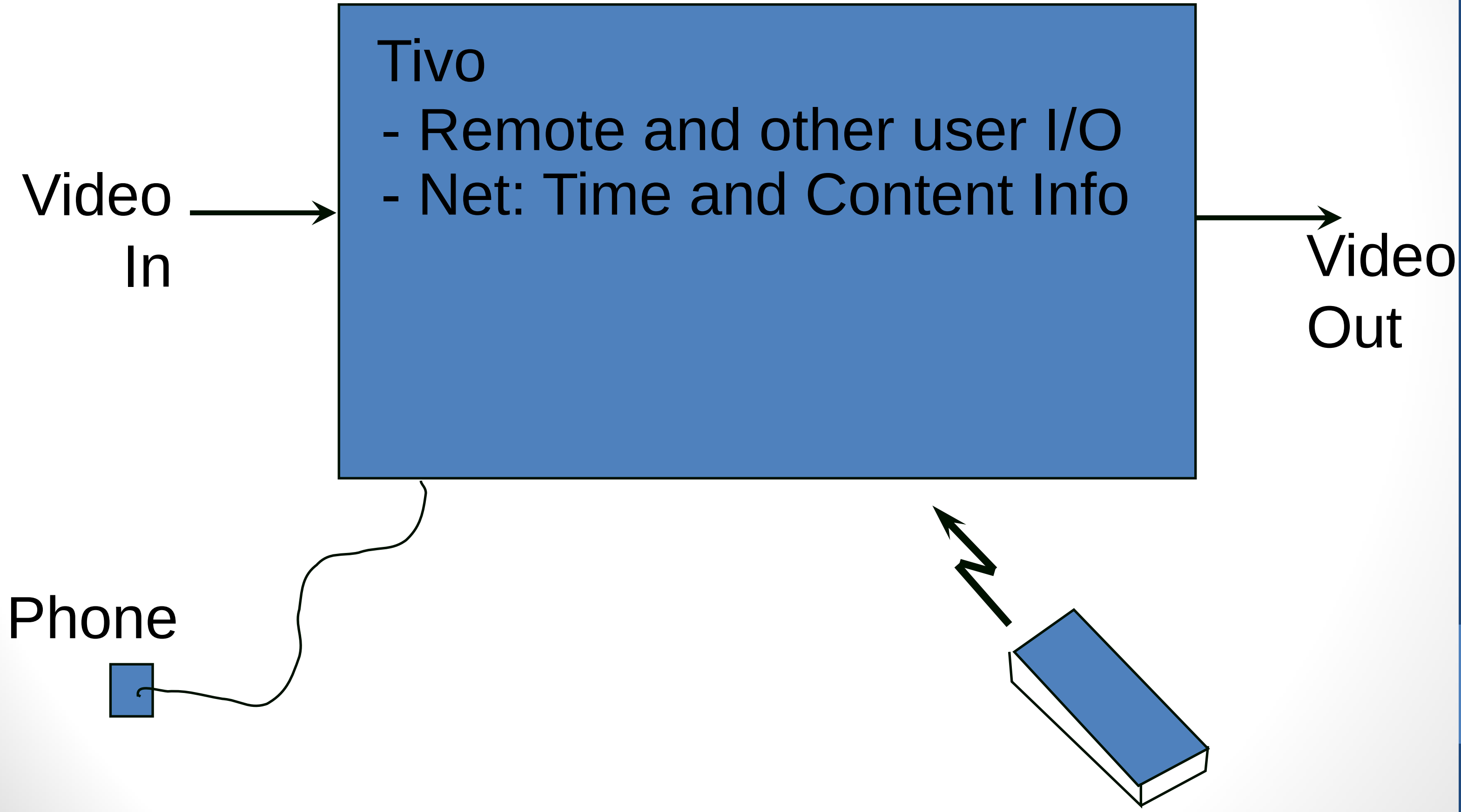
TiVo – Requirements



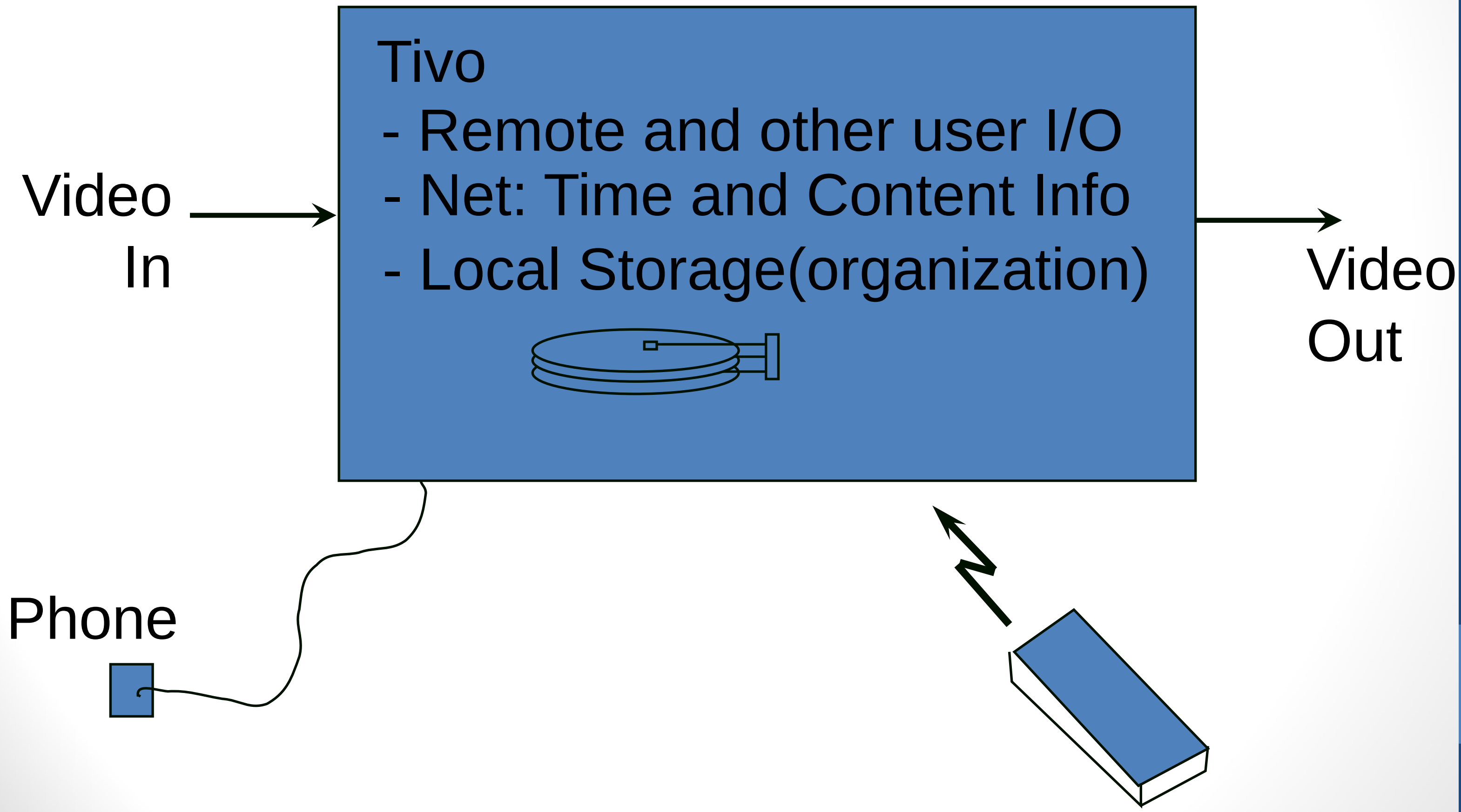
TiVo – Requirements



TiVo – Requirements



TiVo – Requirements


Tivo
- Remote and other user I/O
- Net: Time and Content Info
- Local Storage(organization)
Video In
Video Out
Phone

TiVo – Operating System

- Requirements
- – Video I/O
- – User I/O
- – Networking (on demand)
- – Storage (file system)
  - – Video Data
  - – Content Info
  - – Operating Software

TiVo – Operating System

- Requirements
– Video I/O
– User I/O
– Networking (on demand)
– Storage (file system)
  – Video Data
  – Content Info
  – Operating Software

- Everything required for an operating system

TiVo – Operating System

- Requirements
- – Video I/O
- – User I/O
- – Networking (on demand)
- – Storage (file system)
  - – Video Data
  - – Content Info
  - – Operating Software

- Everything required for an operating system
- – rather than develop from scratch, adopted an existing operating system (Linux)

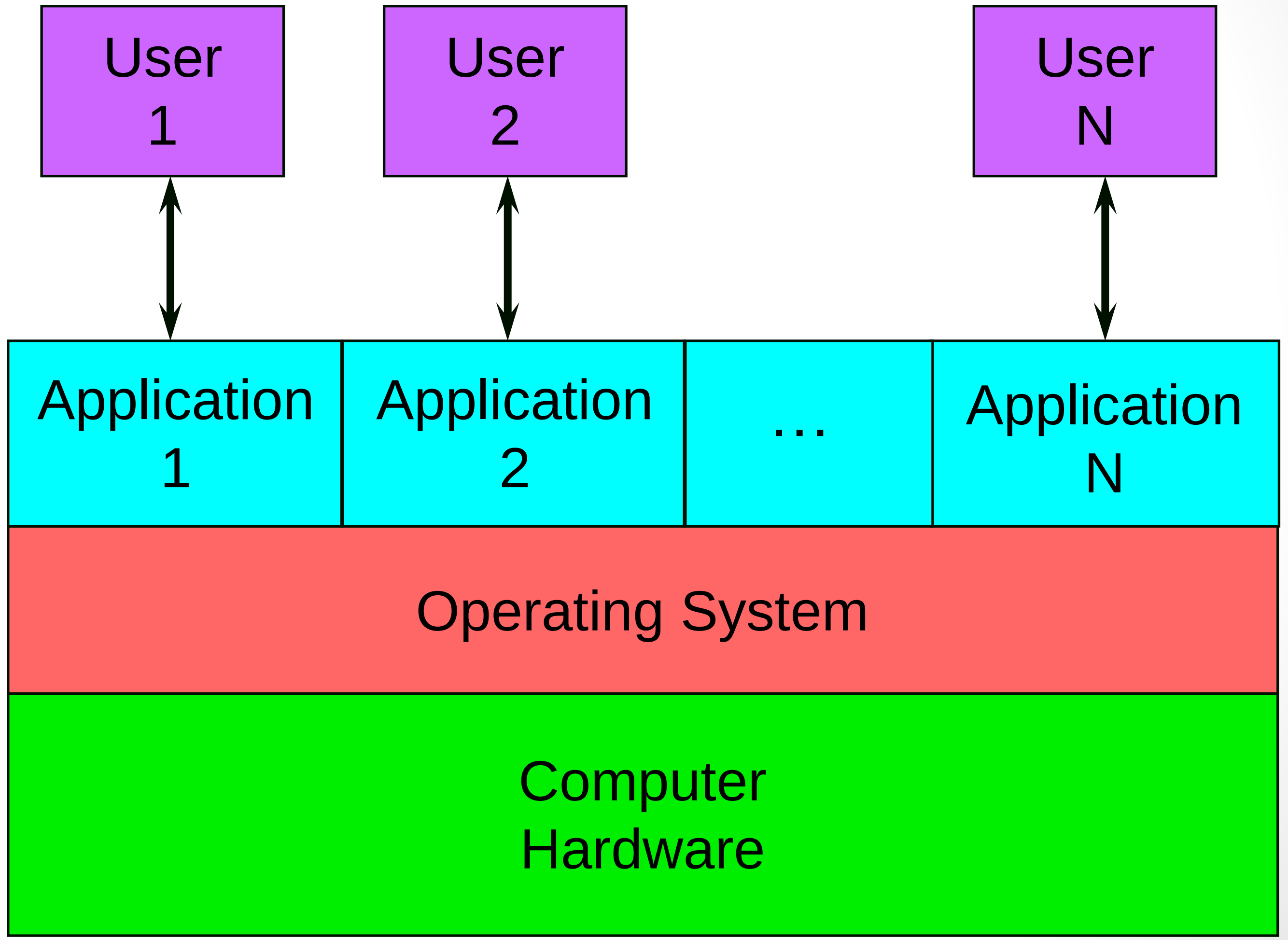
What is an Operating System?

- A Program that acts as an intermediary between a user and the computer hardware

- Goals:
  ◊ Execute user programs and make solving user programs easier
  ◊ Make the computer system more easier to use

- Use the computer hardware in an efficient manner

Computer System Components



What is an Operating System?

Definitions:

- Resource Allocator
  ◊ Manage and allocate resources
  ◊ Provide a uniform interface to similar hardware

- Control Program
  ◊ Controls the execution of user programs
  ◊ Controls operation of I/O devices (i.e. disk)

- Kernel
  ◊ The one program running at all times
  ◊ IPL – Initial Program Load

Single Program Systems

- Early Batch Mainframe Systems

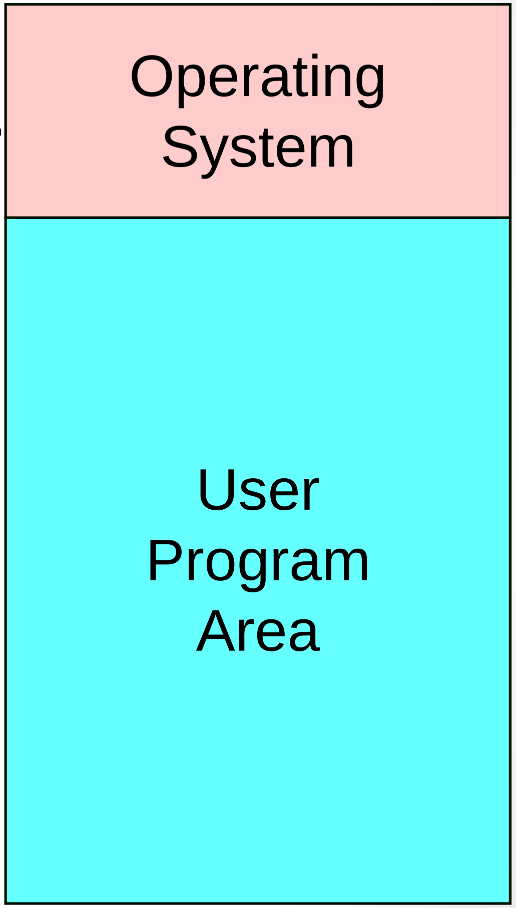
– pool of jobs each run sequentially.

- MS-DOS

- Single Process at a Time
- CPU is idle while waiting for I/O

– wasted cycles

- Minimal hardware/OS requirements



Multiprogram Systems

- Later Batch Mainframe Systems
- Pool of Jobs
  - ◊ More than one program in memory at a time
- When one program is doing I/O, anther gets CPU
  - ◊ Eliminate wasted cycles

- More hardware/OS requirements
  - ◊ OS provides I/O routines
  - ◊ Memory management and security
  - ◊ CPU scheduling
  - ◊ Device Allocation

Time Sharing/Interactive

- CPU is shared between several program kept in memory.
- Programs are swapped between disk and memory.
  ◊ sometimes only parts of programs are swapped.
- May be more than one user

- More hardware/OS requirements
  ◊ Communication with user (terminal, WIMP)
  ◊ Scheduling differences from batch multiprocessing

Parallel Systems

- Tightly coupled
  ◊ processes share memory and I/O
- Advantages
  ◊ Increased throughput
  ◊ Reliability
- Symmetric
  ◊ Each processor identical, process run on whatever processor is available (may run on more than one during it’s lifetime)
- Asymmetric
  ◊ Master–slave relationship

Distributed Systems

- Distribute computation among several processors
- loosely coupled
  - ◊ Each process has own memory, I/O devices
  - ◊ Communication infrastructure
- Advantages
  - ◊ resource sharing/load sharing
  - ◊ reliability

- Client–Server
- Peer–to–Peer
- Clustered Systems

Real–Time Systems

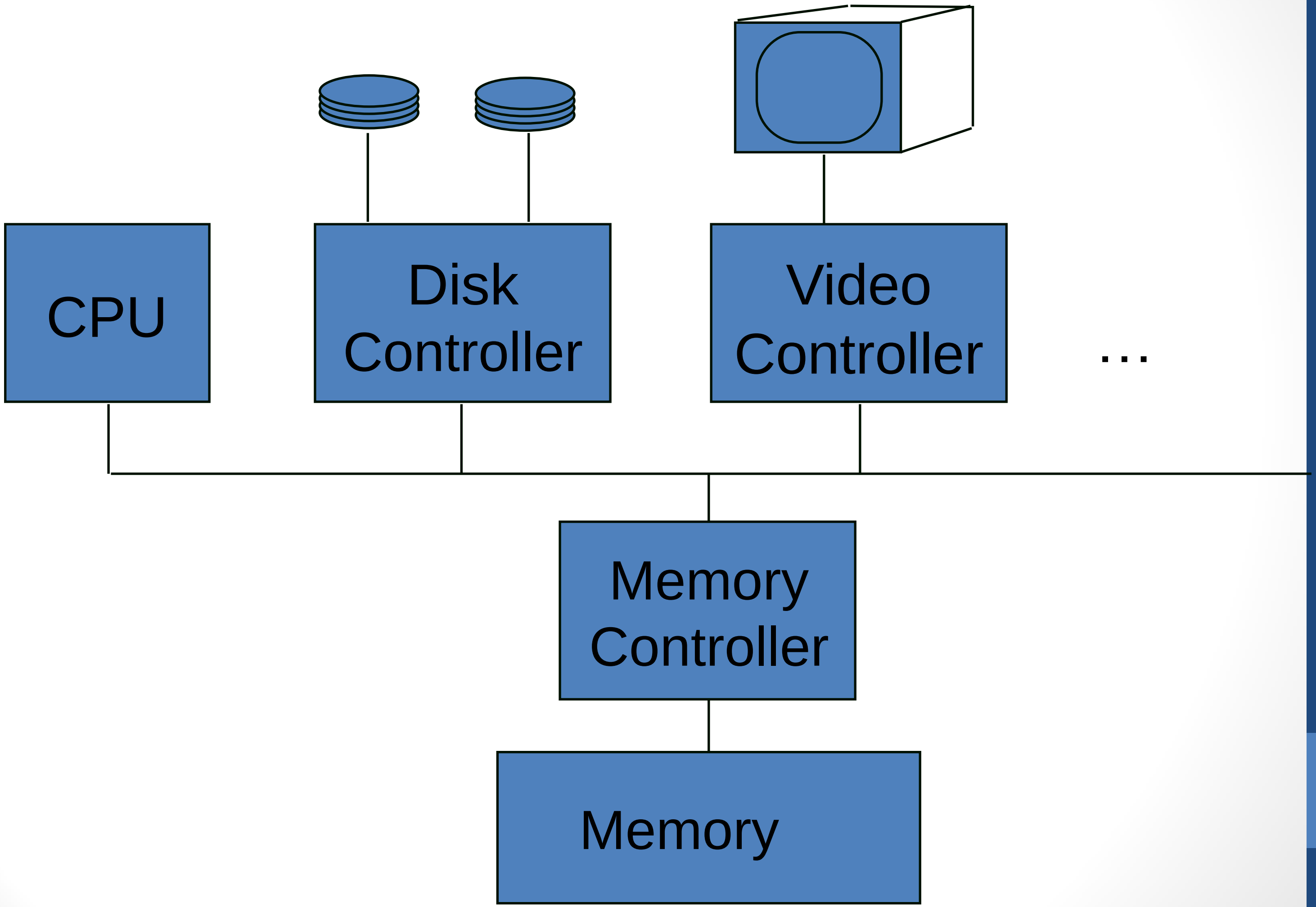
- Usually in embedded systems
- Well–defined fixed time constraints

- Hard real–time
  ◊ hard limits that must be met or the system fails

- Soft real–time
  ◊ A missed deadline does not mean system failure, but value of result decreases with time. Example: Real time video

Computer System Structures

Computer System Structures



Computer System Structure

- CPU and devices execute concurrently
- Each controller is in charge of a one or more devices of a similar type
- Device controller has some sort of local buffer that is used to communicate data to and from the device
- Either CPU or device controller moves data to and from main memory
- Interrupt used to indicate the operation is complete

Interrupts

- Suspends execution of the current program (saves current execution location)
- Transfers control to the interrupt routine
- Determines which controller generated the interrupt
  - ◊ Polling
  - ◊ Vectored Interrupts
- Separate routines for each type of interrupt

Interrupts and Traps

- Interrupts are asynchronous
  ◊ They can happen at any time

- *Traps* are like interrupts, but are generated by the program executing.
  ◊ Division by Zero, other exceptions
  ◊ Trap instruction to request a service from the operating system (Friday's lecture)

- Traps typically use the same mechanism as interrupts for management.

I/O Paradigms

- Synchronous I/O
  - control returns to program only after I/O is complete
  - wait instruction or polling
  - more efficient to give the CPU to another process
  - Common in multi user systems
  - If no process ready to run, the entire system waits.
- Asynchronous
  - return directly to program
  - program registers a notification routine to be notified when the I/O is done (or asks later)
  - Common in single user systems (Windows, Macintosh)

I/O Queues

- More than one process may request I/O from a controller (e.g. more than one program reading a file)

- If the device is busy, then the I/O request is queued. When the interrupt for that device occurs, the next request is removed from the queue and given to the device controller.

- In the example, when one block has been read for one program, the next read request is passed to the disk controller