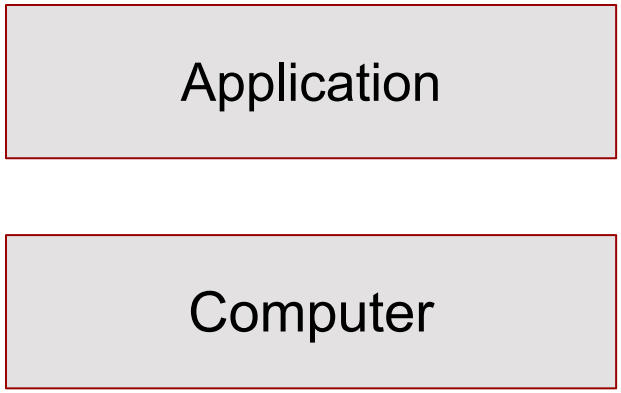
EC 440 – Introduction to Operating Systems

Orran Krieger (BU)
Larry Woodman (Red Hat)

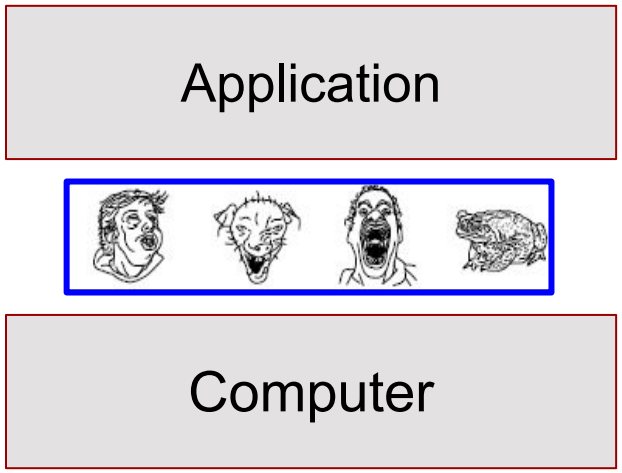
Outline

- What is an operating system
- How we got here
- Our background
- What you will learn & course administration

What is an Operating System?



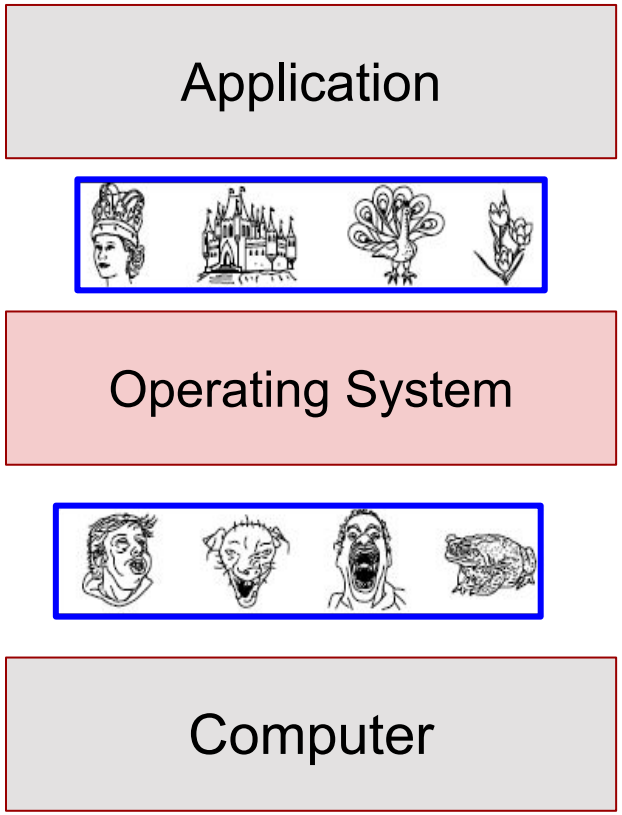
Problem: HW interfaces are ugly



- CPUs
- Memory
- Caches
- Buses
- Storage devices
- Serial ports
- Network interface cards

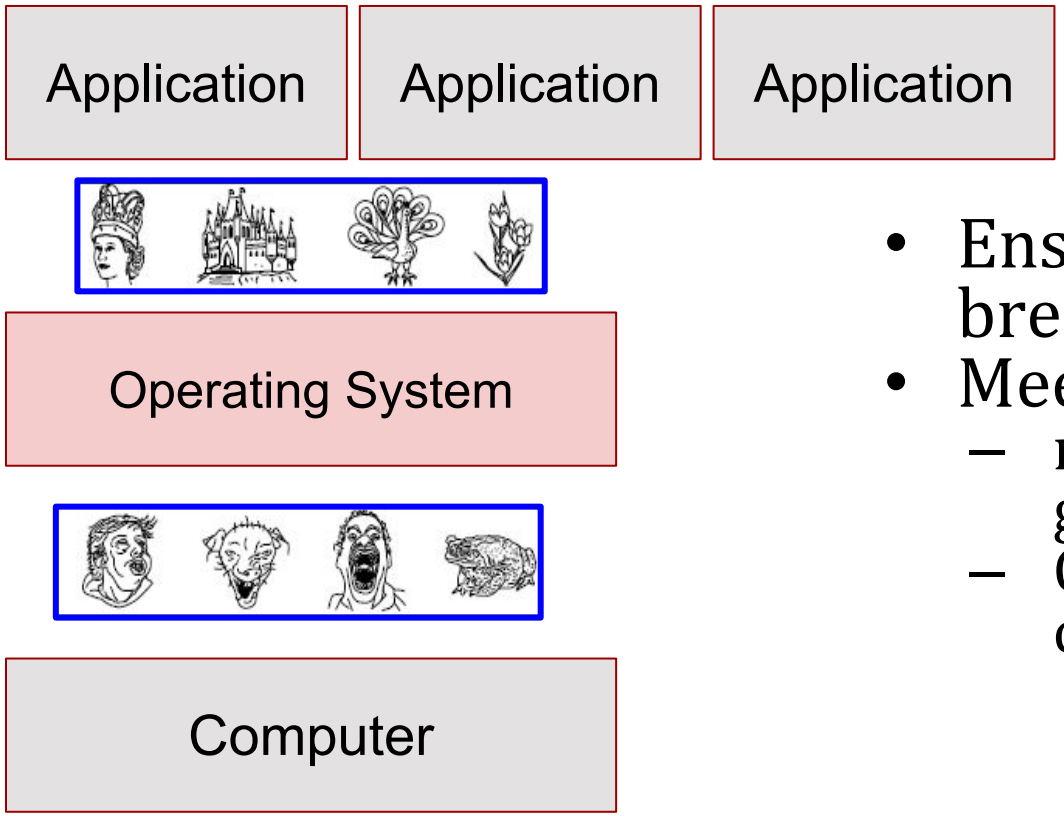
If you needed to understand all the Hardware, no code would ever be written.

At base level, the OS creates clean interfaces



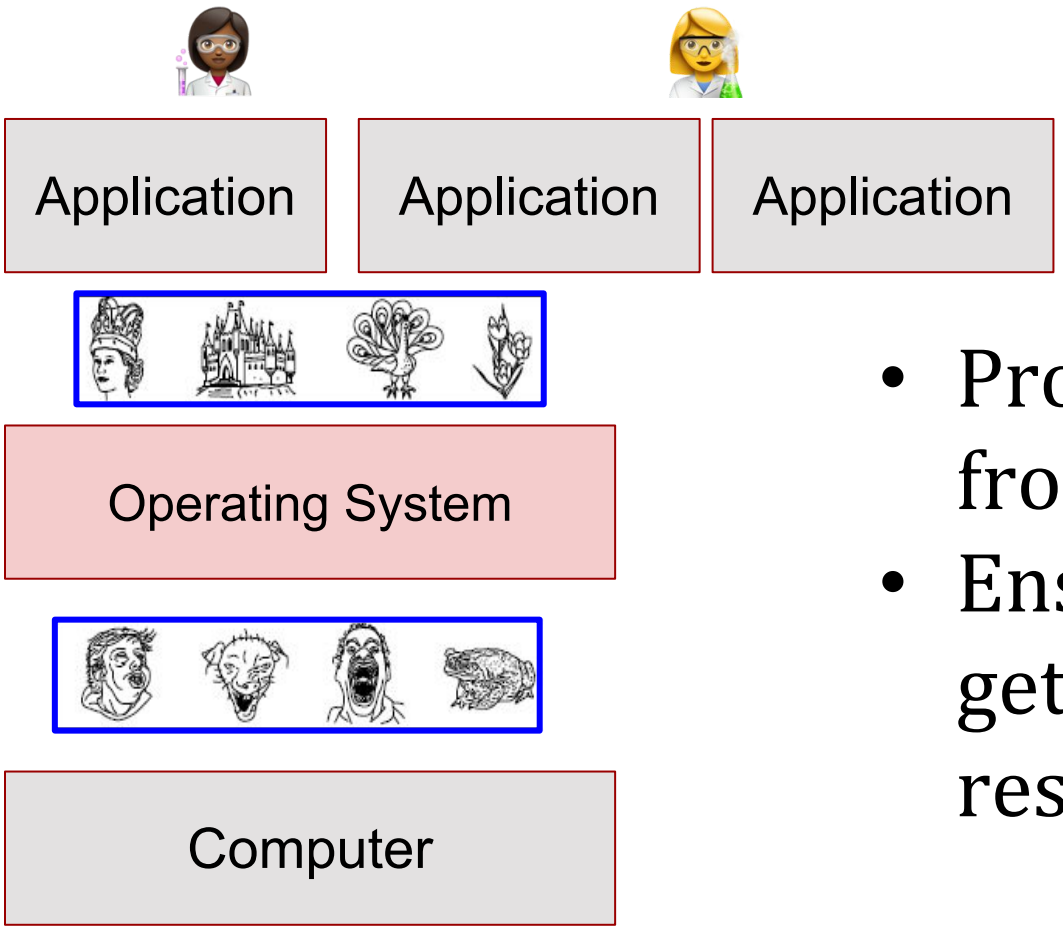
- Processes/threads
- Directories/Files
- Virtual memory
- Sockets
- …

Allow many applications to run



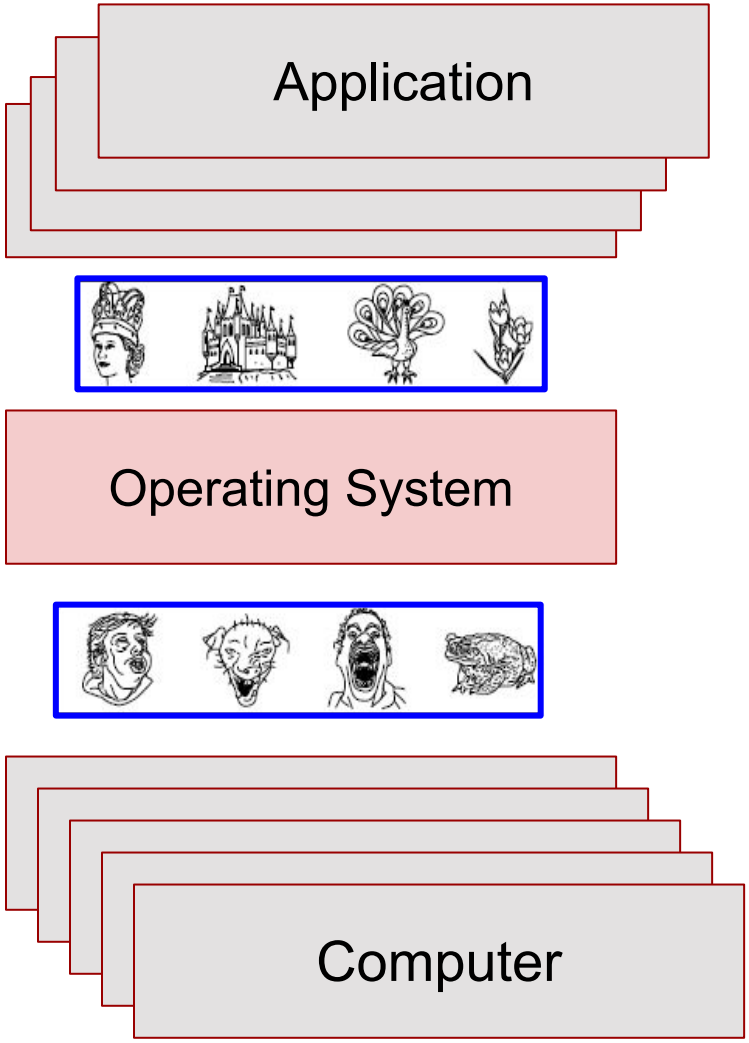
- Ensuring they can't break each other.
- Meeting their needs:
  - real time, e.g., car, gaming...
  - CPU intensive, e.g. game, computational science

From Different Users



- Protecting users from each other
- Ensuring that each gets their fair use of resources

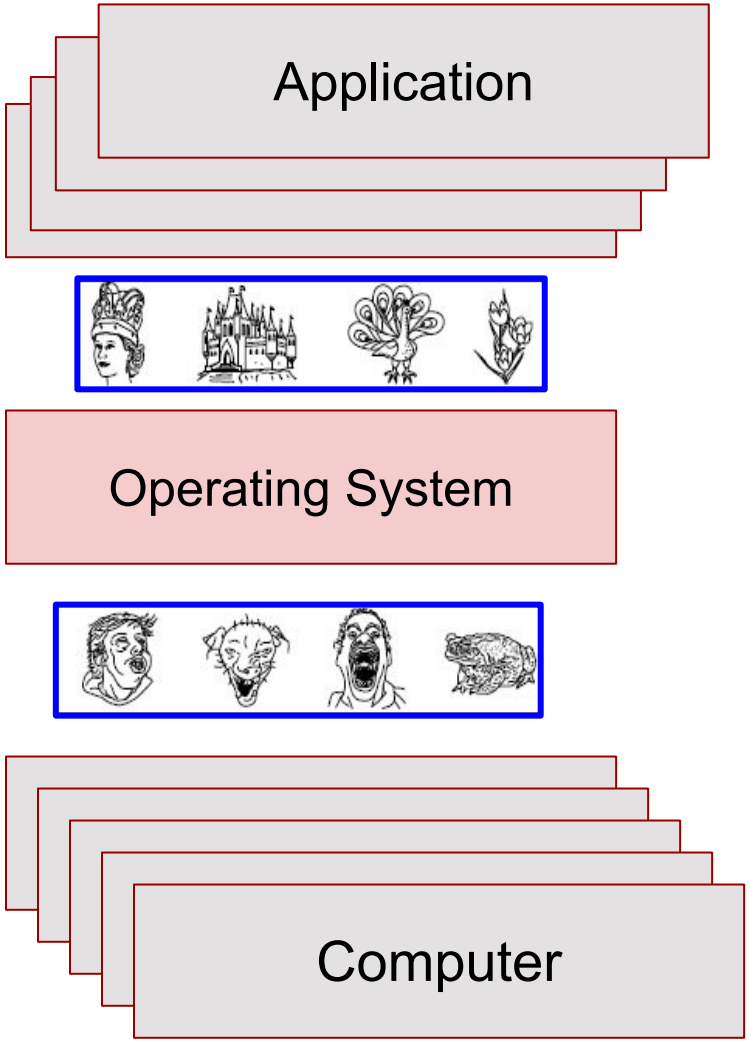
On many different types of computers

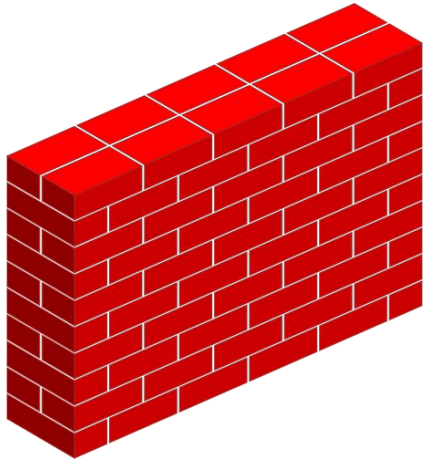


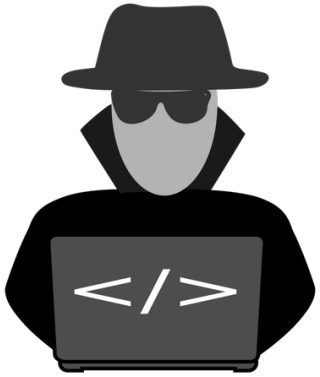
- Processor Architectures: x86, Power, ARM, RISC5...
- Across: PCs, servers, cars, phones, tablets, ...

Enabling common applications and tools...

Protecting against attacks

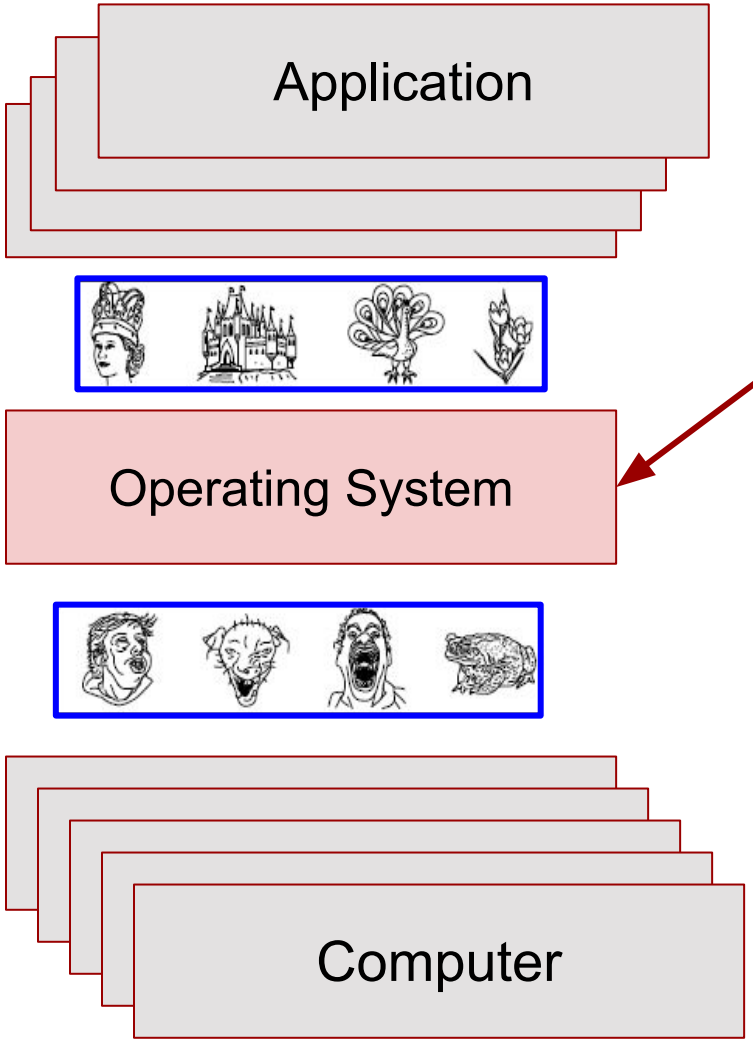


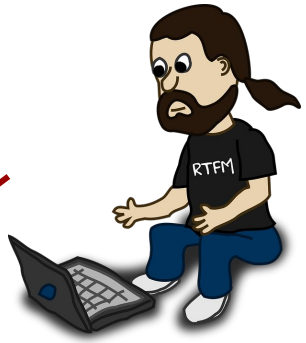




- Firewalls
- Logging
- Security policies

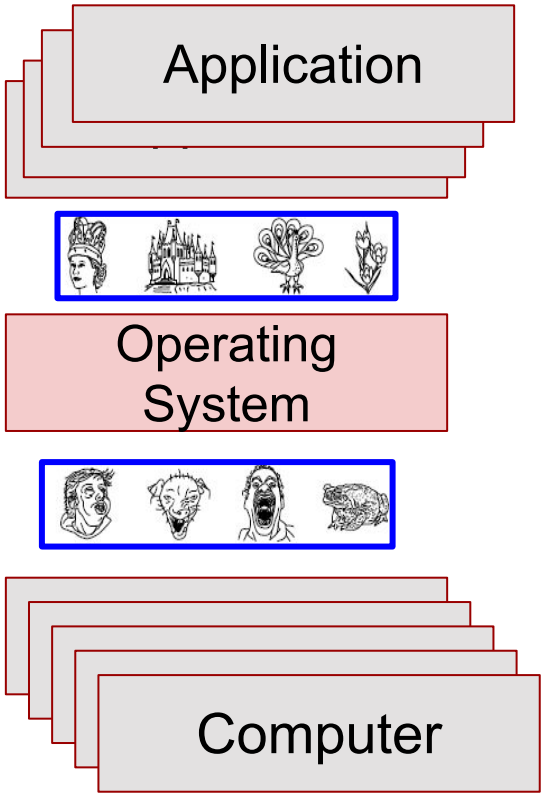
Enabling configuration control, automation, upgrading...



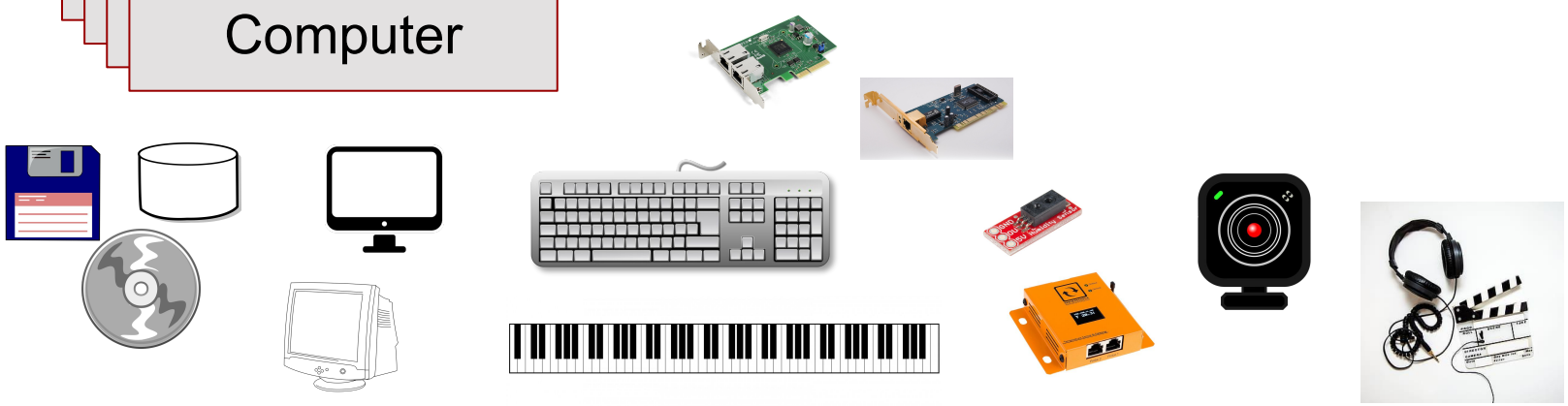


- Personal Computers
- Embedded devices: cars, fridges, routers...
- Massive cloud data centers
- High Performance Computing environments

Enabling devices



- Huge range of devices: disks, monitors, keyboards NICs, sensors...
- Manufacturer needs to work with OS
- OS enabling them to be used across many different computers

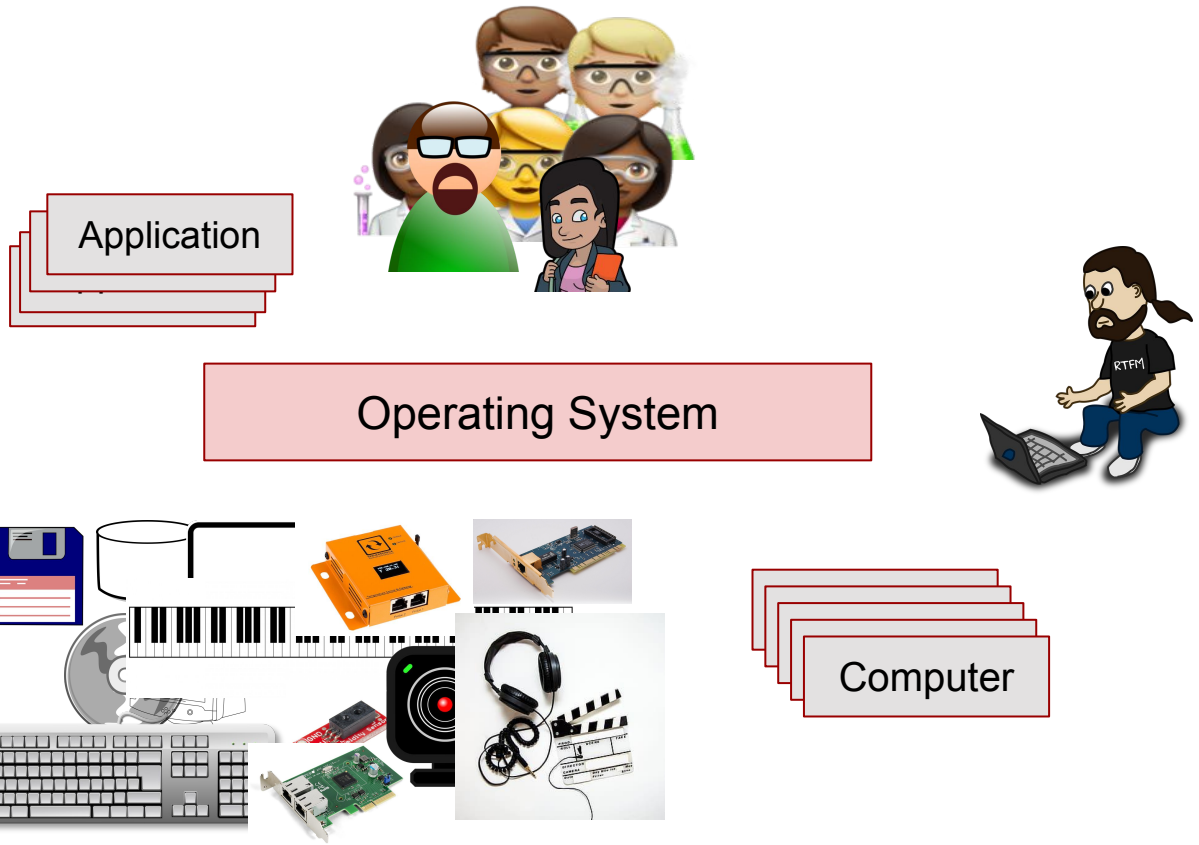


So the job of the modern OS is to

- Create abstractions on computer resources that abstract those resources
- Manage and multiplex the hardware between different application
- Enabling:
  - application to be run on wide set of computers
  - broad set of devices
  - wide range of use cases: phone to cloud
- All while enforcing security

Why do we think is so cool?

Core platform, everything relies in it, if you control the platform you control everything.



This means...

- Focus of some of the largest companies in the world: Microsoft, VMware, Red Hat
- OS people play a key role in every major technology company: Google, Facebook, ..
- Every major change in technology requires OS innovation.
- Led to broader systems, distributed systems: OSDI/SOSP/Usenix are where Borg/Ceph... published
- Started the world of Open Source
- Key to the cloud

Open Source

- Operating systems started the whole move to Open Source...
- We used to think that only cathedral builders could create a stable OS, but Linux showed BAZAAR could work. it is becoming the dominant OS in cloud, mobile devices, enterprise…
  - Keeps any company from owning monopoloy
  - Everyone can scratch their own itch
- Open Source has become requirement for platforms, e.g., HDFS, Hadoop, SPARK, Kubernetis, Ceph...

Outline

- What is an operating system
- How we got here
- Our background
- What you will learn & course administration

History of OS/HW from book

Gen 1: 1945–55:
vacuum tubes

Gen 2: 1955–65:
transistors

Gen 3: 1965–1980:
Integrated cir

Gen 4: 1980– : PCs

Gen 5: 1990– :
Mobile

Gen 6: 2015- : Cloud

Generation 1: Vacuum tubes

Gen 1: 1940–55: vacuum tubes
Gen 2: 1955–65: transistors
Gen 3: 1965–1980: Integrated cir
Gen 4: 1980– : PCs
Gen 5: 1990– : Mobile
Gen 6: 2015- : Cloud

- No operating system: all programming done in machine language, or plugboard
- One program & one user at a time on the computer
- Computers special purpose, size of rooms
- Watson, 1945, worldwide market for maybe 5 computers

Generation 2: Transistors & Batch

Gen 1: 1945–55: vacuum tubes
Gen 2: 1955–65: transistors
Gen 3: 1965–1980: Integrated cir
Gen 4: 1980– : PCs
Gen 5: 1990– : Mobile
Gen 6: 2015- : Cloud

- Rise of the mainframe that can be sold to companies
- Only interfaces are pull card deck (fortran) into machine and dump results
- Can quickly run programs one after the other
- First OSes:
  - Job control language
  - Compiler for Fortran

Generation 3: Transistors & Batch

Gen 1: 1945–55: vacuum tubes
Gen 2: 1955–65: transistors
Gen 3: 1965–1980: Integrated cir
Gen 4: 1980– : PCs
Gen 5: 1990– : Mobile
Gen 6: 2015- : Cloud

- First OS that runs across multiple lines (IBM 360)
- Spooling
- Multiprogramming
- Timesharing
- Virtualization
- MULTICS: Utility computing, hierarchical file system, segments...
- Unix: single-user version of MULTICS
  - MacOS, Linux (Android)
  - Open Source Key success

Generation 4: Personal Computers

Gen 1: 1945–55: vacuum tubes
Gen 2: 1955–65: transistors
Gen 3: 1965–1980: Integrated cir
Gen 4: 1980– : PCs
Gen 5: 1990– : Mobile
Gen 6: 2015- : Cloud

- Wide variety of OSes for hobbyist... then IBM decides to create PC
- MS-DOS creates massive monopoly, first to recognize OS as control point
- Eventually MAC adopts BSD (& microkernel), BSD license

Networks became ubiquitous, then internet...

Generation 5: Mobile

Gen 1: 1945–55: vacuum tubes
Gen 2: 1955–65: transistors
Gen 3: 1965–1980: Integrated cir
Gen 4: 1980– : PCs
Gen 5: 1990– : Mobile
Gen 6: 2015- : Cloud

- A new fight for the winning OS... started in 1997
- Now two main competitors, Google's Android which is Linux & iOS
- PCs increasingly irrelevant; world moving to mobile devices and cloud

Generation 6: Cloud

Gen 1: 1945–55: vacuum tubes
Gen 2: 1955–65: transistors
Gen 3: 1965–1980: Integrated cir
Gen 4: 1980– : PCs
Gen 5: 1990– : Mobile
Gen 6: 2015- : Cloud

- In late 90s, Virtualization made a comback with VMware (more later)
- Suddenly computing became a fungible commodity.
- AWS creates a cloud using Xen; open source virtualization based on Linux

Is the fun over?

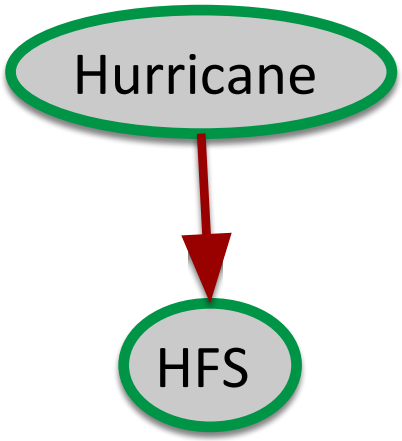
There are fundamental changes in usage and technology:

- Cloud pushing new models of OS for SaaS/FaaS
- Fundamental change in security requirements; provider/tenant
- Denard scaling is over; but network and storage keep getting faster…
- Edge & Cloudlets driven by 5G
- First generation of persistent memory last year
- Rise of smart NICs
- 100 GB networks spanning the world..
- Rise of AI...
- ….

All these require fundamental OS innovation, e.g.: unikernel, partitioning HW, AI controlled policy…

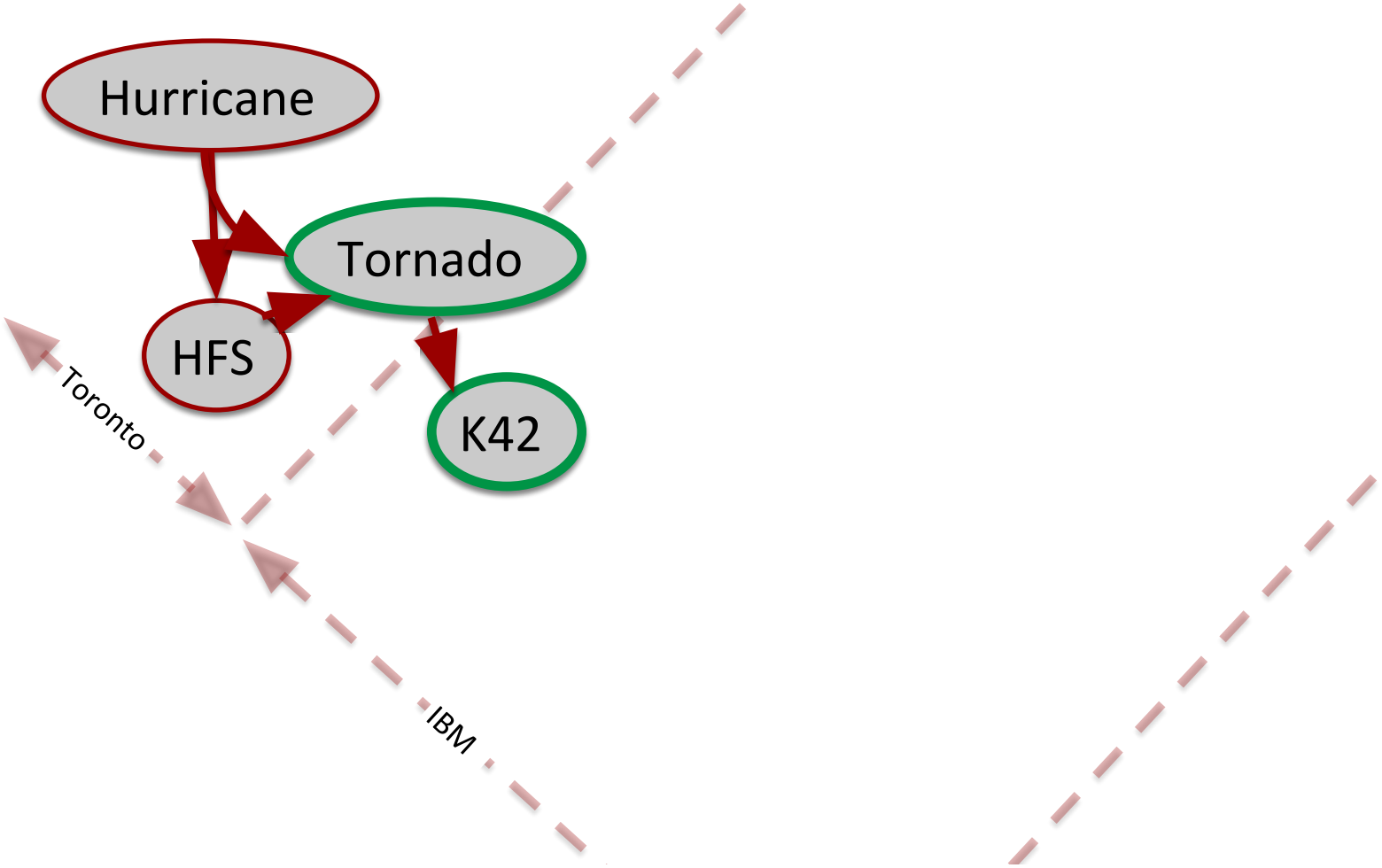
Outline

- What is an operating system
- How we got here
- Our background
- What you will learn & course administration

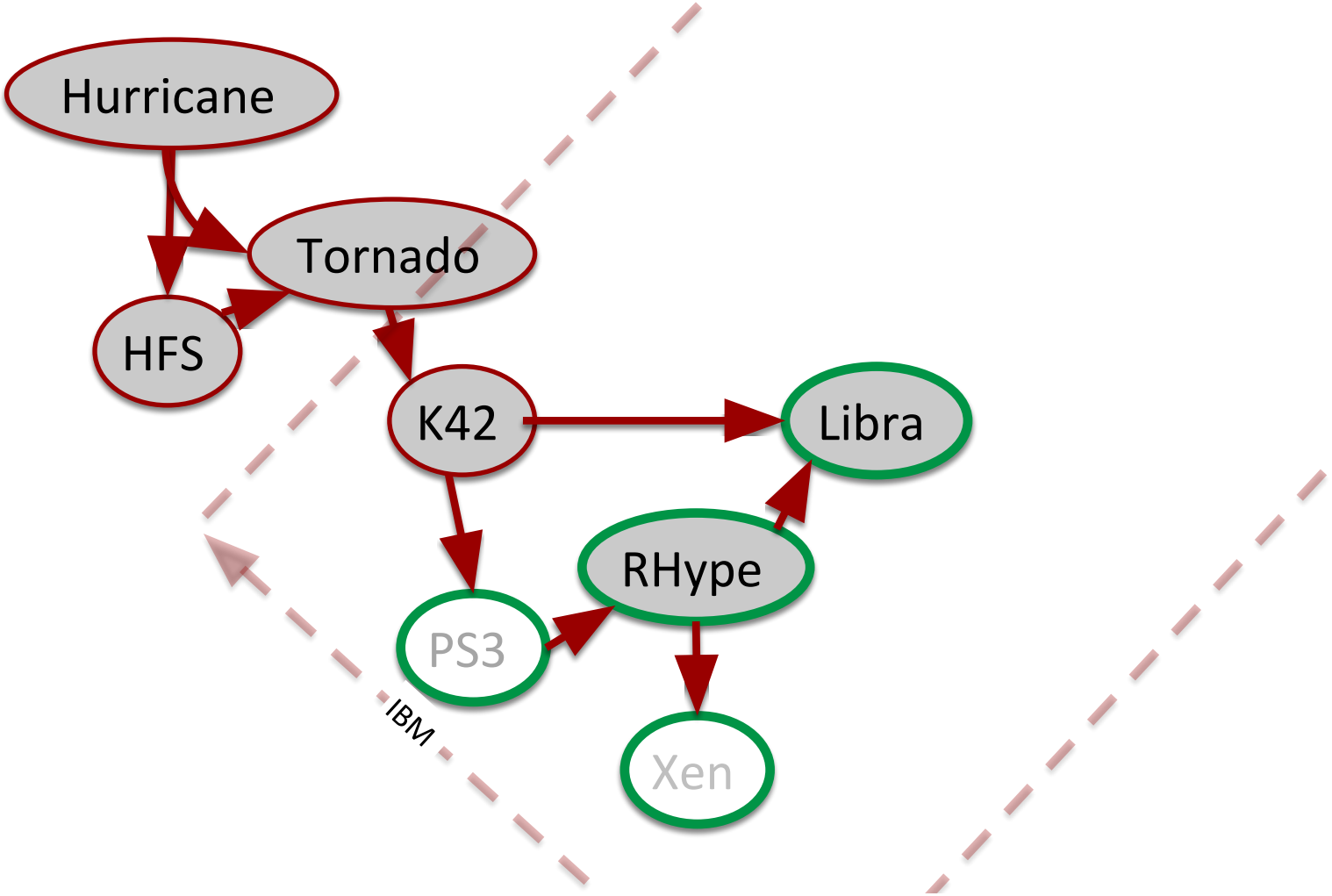


Orran's background

Large-scale NUMA shared memory multiprocessors will become important and current ad hoc approach scalability won’t work

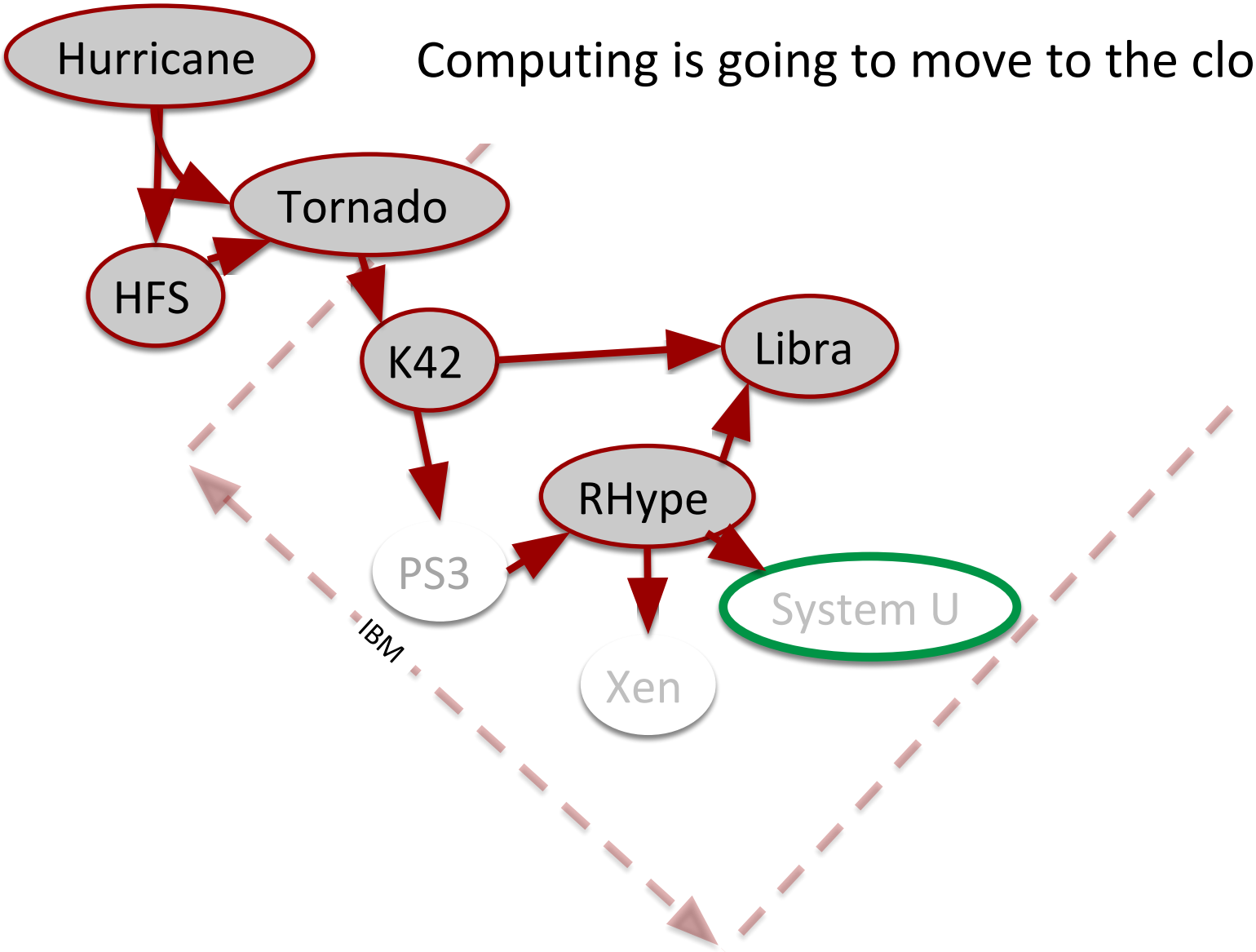


Take 2: Large-scale **64-bit** NUMA shared memory multiprocessors will become important and current ad hoc approach scalability won't work. Customizability key to performance

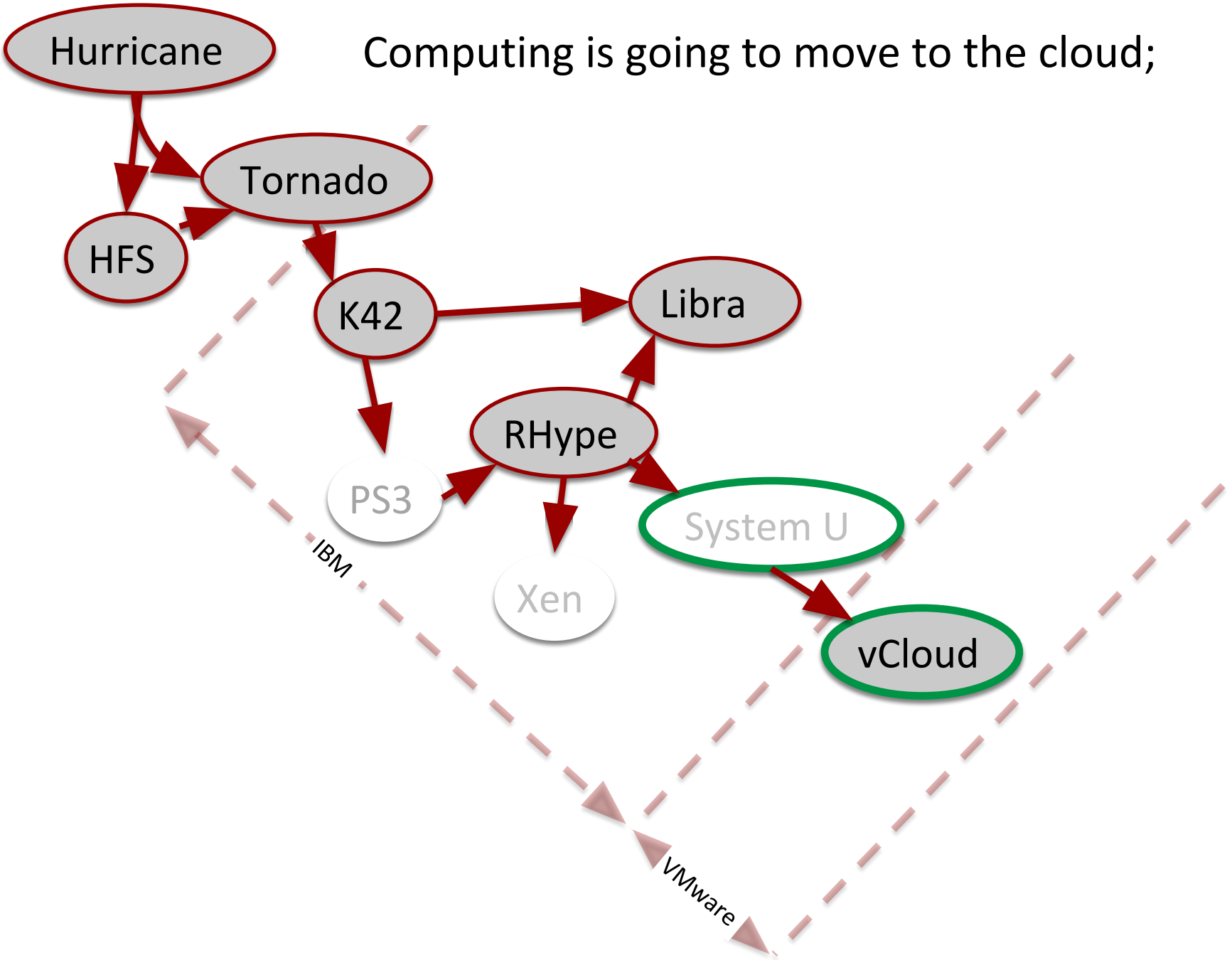


Virtualization adds fundamental level of indirection: deterministic performance, HPC, library Oses for applications, for devices…

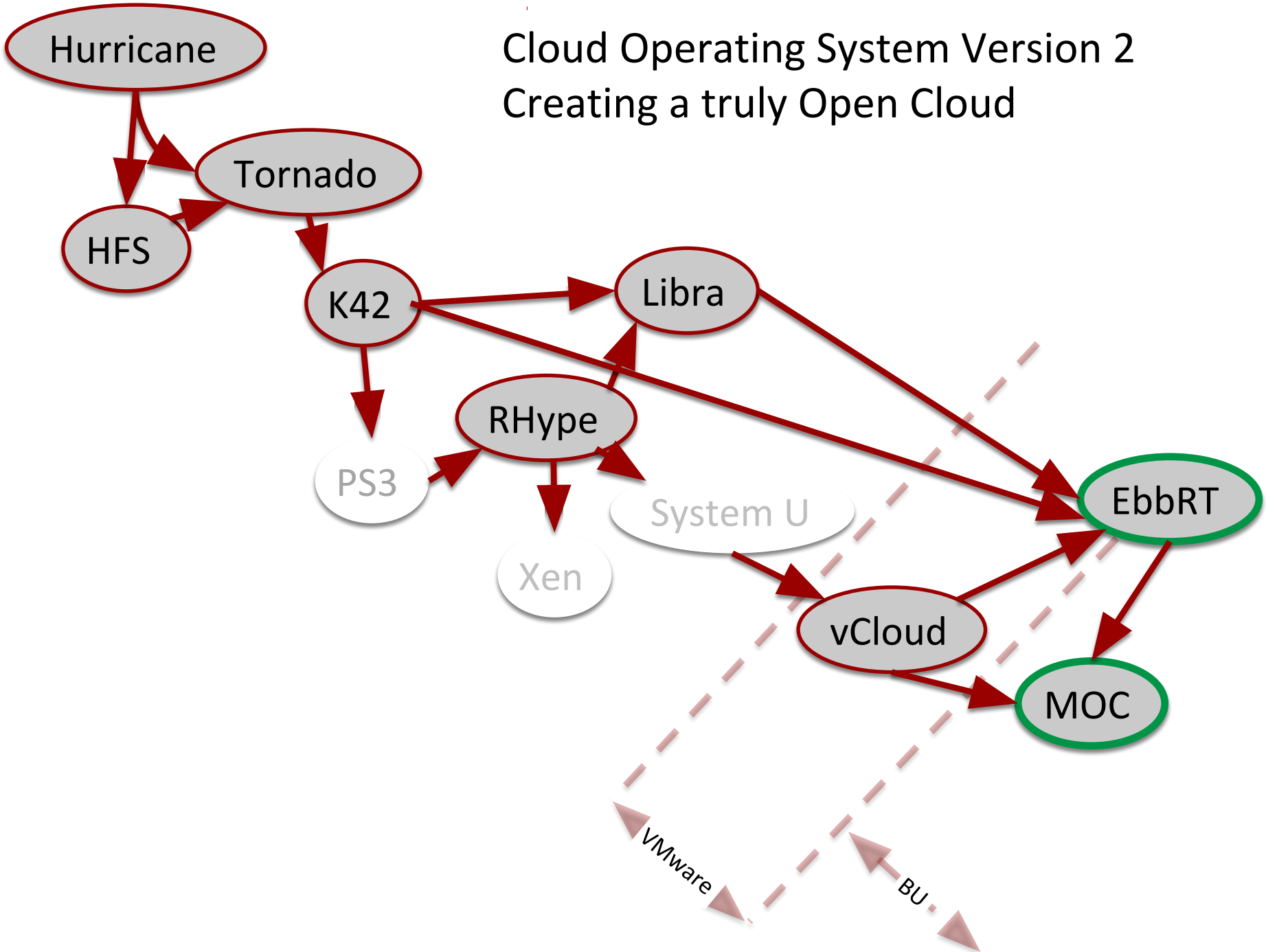
Computing is going to move to the cloud



Computing is going to move to the cloud;



Cloud Operating System Version 2
Creating a truly Open Cloud



Larry's background

- 40 years OS development at AT&T Bell Labs, Sony, Wang Labs, DEC, Red Hat
- Systems: AT&T System V Unix, Wang VSOS & Wang Unix, DEC VMS & Ultrix/Tru64 Unix, Red Hat Enterprise Linux(RHEL)
- Leading OS directions in the Red Hat Collaboratory@BU

Larry's background(cont)

- Larry’s primary kernel work has been:
  - memory management.
  - process control and scheduling.
  - interprocess communication and synchronization.
  - SMP and NUMA support.
  - crash dump support.
  - various file systems.

Larry's background(cont)

- Open source kernel development
  - Years ago every company had their on OS
  - Today most OS development is done via the open source model.
    - large email lists to communicate design and implementation details.
    - Standardized design and coding styles.
    - designed by large committee of software and hardware engineers.
    - Lots of engineers contribute.

What we are working on now

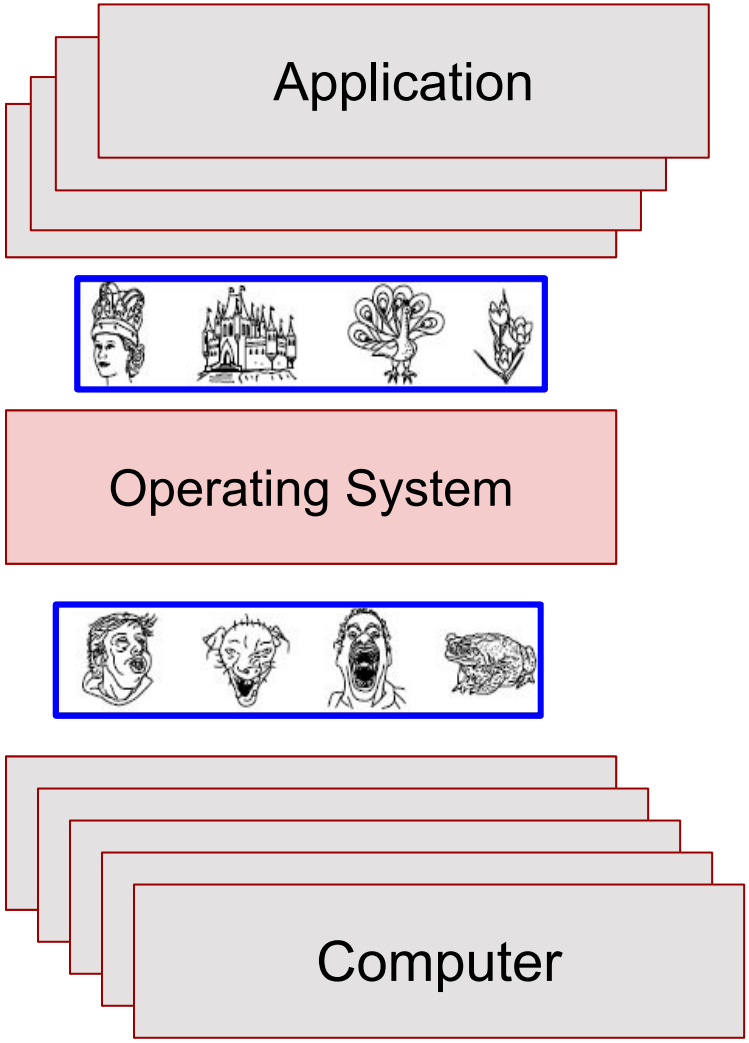
- Red Hat created Collaboratory@BU 4 years ago, we are working on:
  - Unikernel
  - Deterministic performance
  - OS security
  - AI control of OS policy
  - Partitioning Hypervisor
  - Automatic tuning of kernel parameters

- RH/IBM/BU Collaboration to create self-driven open platform from edge to cloud

Outline

- What is an operating system
- How we got here
- Our background
- What you will learn & course administration

Back to OS



Multiplexing

- creating an illusion of multiple (logical) resources from a single (physical) one

Allocation

- keep track of who has the right to use what

Transformation

- creating a new resource (logical) from existing resource (physical)

primarily for "ease of use"

An OS multiplexes, allocates, and transforms HW resources

Types of Multiplexing

Time multiplexing

- time-sharing
- scheduling a serially-reusable resource among several users
  - e.g., CPU or printer

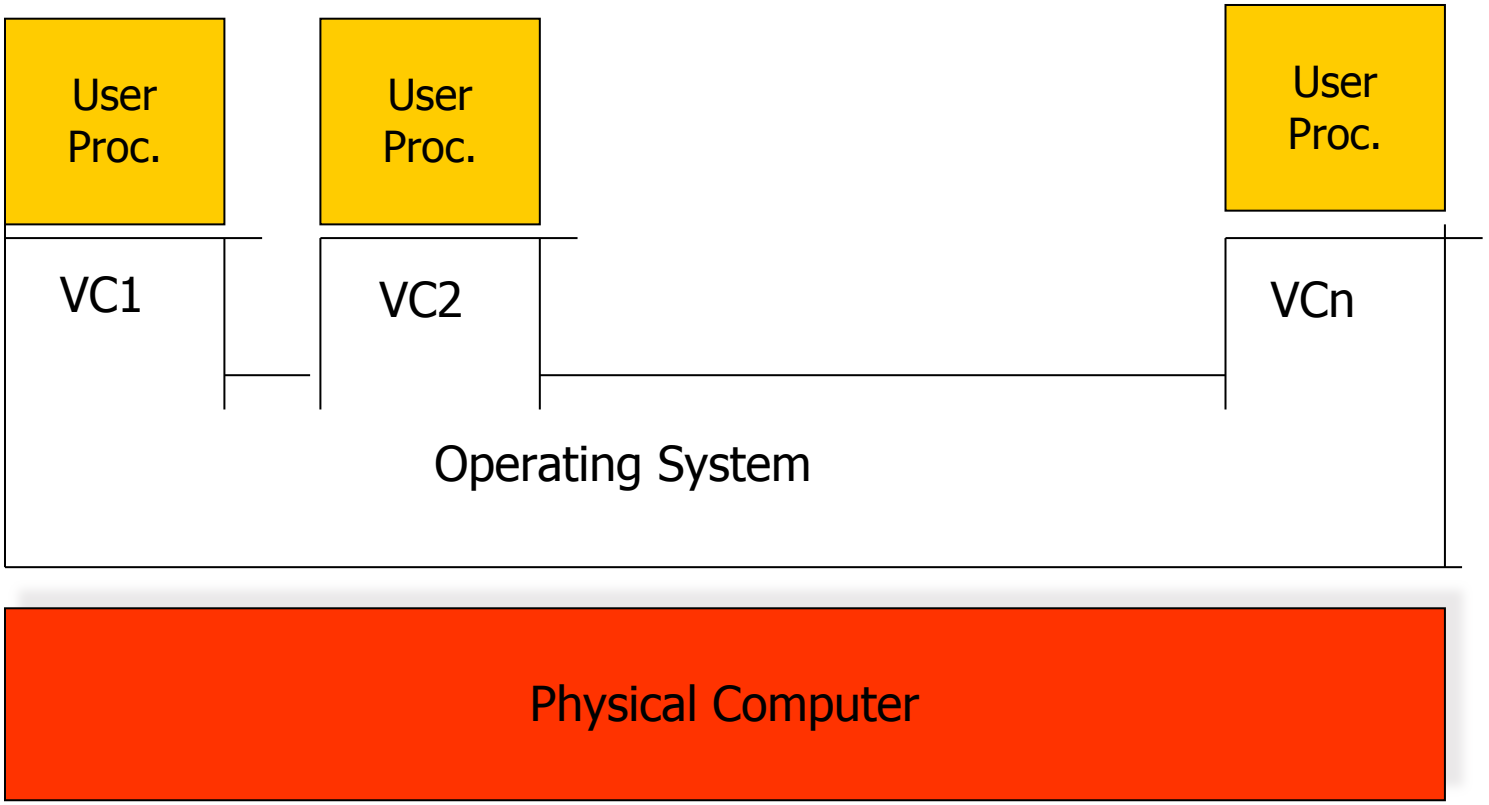
Space multiplexing

- space-sharing
- dividing a multiple-use resource up among several users
  - e.g., memory, disk space

We normally do both

- e.g., memory

Virtual Computers



OS Interface – Virtual Processors

Nearly the same interface as the physical CPU

OS removes privileged operations

- PSW determines if the code is either “user code” or “OS code”
- changes in status are strictly regulated...

OS adds instructions (system calls)

- create new virtual computers (processes)
- communicate with other VCs
- allocate memory
- perform input and output operations I/O
- access the file system

OS Interface – Virtual Memory

- **Memory of the Virtual Computer is similar to the hardware memory (i.e., a sequence of words), and it is accessed the same way**
- **The OS divides up the memory into parts and gives each part to each virtual computer**
- **OS creates an illusion that each virtual computer has a memory starting from address 0x0000**
- **OS creates an illusion that the virtual computer has more memory than the physical memory**
- **The OS isolates and protects the memory between virtual computers**

OS Interface – Virtual File System

- **Secondary storage provides long-term storage for data**
- **Storage is done physically in term of disk sectors and virtually in terms of files**
- **The virtual computer sees a file system consisting of named files with arbitrary size**

OS Interface – Virtual I/O

- **I/O operations of the virtual computer are completely different from the I/O operations of the physical computer**
- **The physical computer has devices with complex control and status registers**
- **In contrast, the virtual I/O is simple and easy to use**
- **In fact, in most OSes (e.g., UNIX) virtual I/O abstraction is almost identical to the file-system interface giving rise to uniformity of treatment with respect to all types of I/O devices including disks, terminals, printers, network connections**
- **Each VC sees a dedicated I/O device: the actual hardware is space/time multiplexed by the OS**

Key OS (e.g., Linux Kernel) services & interfaces

Services

- System HW support
- CPU scheduling
- Memory management
- Device management
- Networking
- File systems
- Process environment

Interfaces

- Configuration tools
- Processes & threads
- Virtual memory
- Device Drivers
- Sockets
- Directories and files
- System calls

What you will learn

- How to write good system's code
- Key OS abstractions and interfaces
- A bit on the HW we are abstracting
- Processes/threads scheduling and synchronization
- Memory Management
- File systems
- Input and Output
- Virtualization and Cloud
- Security

Projects

Programming assignments

1) **Shell (system calls)**
2) **Threads (parallel execution, scheduling)**
3) **Synchronization (semaphores, …)**
4) **Memory (virtual memory, shared regions, …)**
5) **File systems**

We will also give detailed directions to download, build, boot and debug your own Linux kernel, and hope to provide a variant on 5, where you will build your own kernel file system.

- **These are reputed to be really hard! Assume 20-30 hours of work for each.**
- **Don't CHEAT, you can talk about ideas, but no sharing code; you will need to fully explain your solution.**

EC 440 – Course Staff

- Orran Krieger & Larry Woodman
- Daniel Wilson (GTF)
- Jonathan Cameron & Ryan Sullivan (UTF)
- You - help each other, this course is tough… we will keep track of good answers on piazza

Should we have office hours?

EC 440 Info

Main resource: Piazza

- 1st stop for questions (check for duplicates!)
- Help each other with answers (**not solutions!**)
- Resources (lecture slides, etc.)
- Projects/homeworks/challenges will be posted there too
- Class Vidios will be posted on blackboard

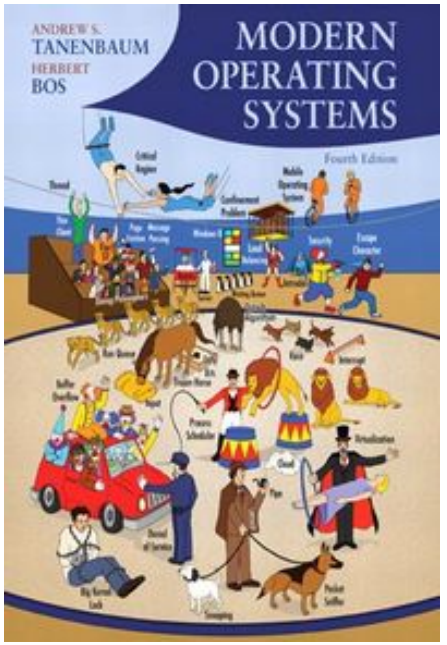
Background

- This seems to be the “killer” course in CE
- We believe you should all be able to succeed… if you put in the work.
- Changing a few things from the past:
  - Handed out pre-challenge to help refresh you on programming in “c” and setting up your environment.
  - Developed a consistent environment with autograder
  - Next lecture Daniel will talk about programming
- We believe that you can do this, but we are not going to make it easier...

Material

We will be informally using:

- Andrew S. Tanenbaum and Herbert Bos, Modern Operating Systems, Prentice-Hall, 2015



How we will mark

- Lots on the challenges ~75
- A fair bit on the exams ~20
- Some on the discussions on Piazza… ~5

We might add some bonus challenges, still thinking about it...