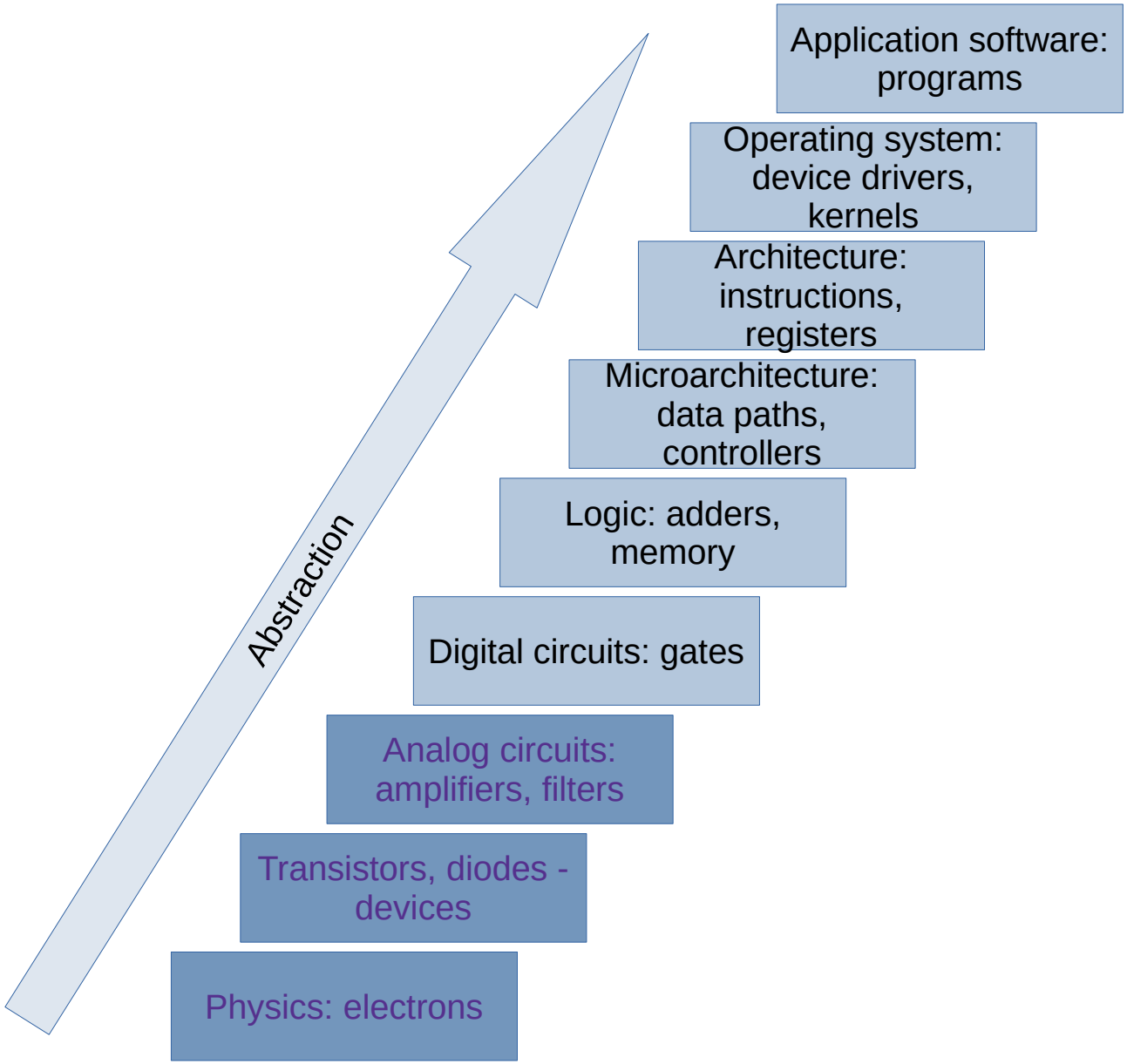
E2011: Theoretical fundamentals of computer science
Basic concepts about operating systems

Vlad Popovici, Ph.D.

Fac. of Science - RECETOX

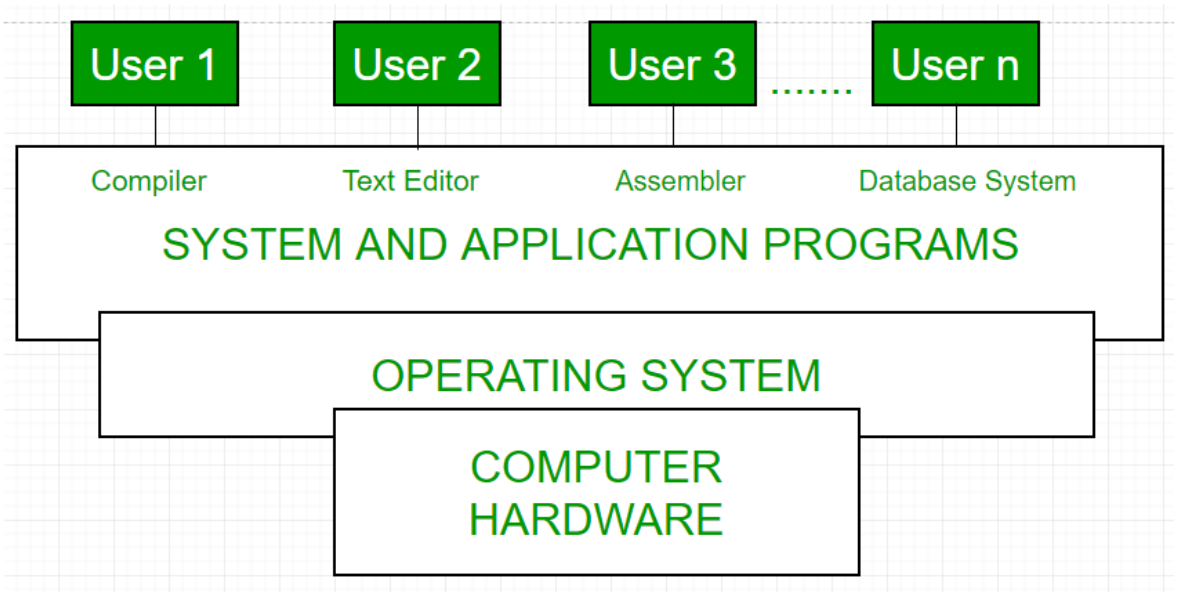
Outline


Application software: programs
Operating system: device drivers, kernels
Architecture: instructions, registers
Microarchitecture: data paths, controllers
Logic: adders, memory
Digital circuits: gates
Analog circuits: amplifiers, filters
Transistors, diodes - devices
Physics: electrons
Abstraction

Operating systems

Why?

- acts as an interface between user/applications and hardware
- resource manager: manages I/O and peripherals
- provide a *virtual perspective* on the underlying hardware
- manage programs



Main roles of the OS:

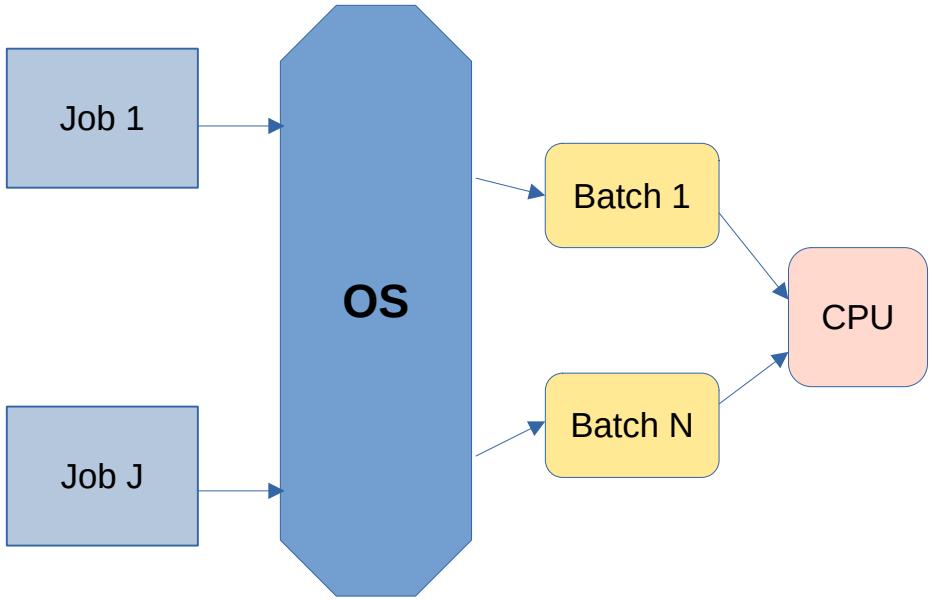
- *resource sharing*:
  - *allocate* resources for all activities; separate the resources between activities
  - *isolation* of activties
  - *communication* between processes/activities
- *virtualization*
- provide *standard services*: process management, file systems, network services, etc

OS over time

- early OS: set of routines for common procedures; single-user OS
- multi-user OS: $\rightarrow$ batch processing $\rightarrow$ multi-tasking $\rightarrow$ virtual machines
- time-sharing OS: interactive use
- modern OS: usually a kernel+GUI; desktop-level (MacOS, Linux, Windows, etc); smart appliances (Android, iOS, Symbian, etc); server-level: GUI is optional (UNIX-based OSes, etc); etc

Batch OS

- intially, a system in which *jobs* are run sequentially
- a *job monitor* supervises execution and manages the *job queue*
- modern OSes have their own *job scheduler(s)* allowing for non-interactive, eventually synchronized, execution of jobs



Example of batch OS

IBM's

OS/360

- developed in the 1960s
- designed for mainframes
- goal was to maximize hardware utilization
- introduces *job control language (JCL)*

Examples:

- start a job: job name, accounting information, params
  ```
  //MYJOB JOB (ACCOUNT),'MY JOB NAME',CLASS=A,MSGCLASS=X
  ```
- specify program to execute within job
  ```
  //STEP1 EXEC PGM=MYPROGRAM
  ```
- conditional execution
  ```
  //STEP2 EXEC PGM=ANOTHERPGM,COND=(0,LT,STEP1)
  ```

Multitasking

- "concurrent" exectution of *processes* (*tasks*)
- does not imply parallel execution
- *multiprogramming OSes*: allow context switching between processes
- *cooperative multitasking*: processes voluntarily ceed time to OS/other process: ealry Windows and MacOS
- *preemptive multitasking*: OS decides to switch between executing tasks
- *real time* systems
- *multi-threaded systems*

Example of multitasking OS

UNIX

- developed in the 1970s at Bell Labs
- implements preemptive multitasking
- time-sharing: CPU time dived among multiple processes
- process management
- system calls for process control: `fork`, `exec`, `kill`, etc

Examples:

- work with files: `ls`, `cp`, `mv`, `rm`
- traverse filesystem hierarchy: `cd`, `pwd`
- control access: `chmod`, `chown`
- control processes: `ps`, `kill`

Another classification of modern OSes

1. **Desktop OSes:** for personal computers, user-friendly, wide range of applications; ex: Windows, macOS, Linux
2. **Mobile OSes:** for touch interfaces and mobile devices; ex: Android, iOS, HarmonyOS.
3. **Server OSes:** manage and optimize network resources, security, and multi-user services; ex: Windows Server, Linux
4. **Embedded OSes:** for specific hardware and applications; ex: FreeRTOS, Embedded Linux, VxWorks, QNX.
5. **Real-Time OSes:** when immediate response to inputs is critical; ex: RTLinux, FreeRTOS, QNX, LynxOS.
6. **Mainframe OSes:** for high-volume processing; ex: IBM z/OS, Unisys OS 2200, HP NonStop OS.
7. **Distributed OSes:** coordinates multiple computers ex: Apache Hadoop (for data processing clusters), Google's Kubernetes OS (for container management), Plan 9 from Bell Labs.
8. **Network OSes:** used for managing networked computers; ex: Novell NetWare, Cisco IOS, Windows Server with Active Directory.

OS Kernel

- *kernel*: the core of the OS that provides services to all other components of OS
- talks directly to hardware
- usual start-up sequence: power-on $\rightarrow$ BIOS (Basic I/O System) $\rightarrow$ kernel loaded into a protected memory space

OS Kernel - main functions

- loading and managing less-critical OS components, such as device drivers
- managing execution threads and various processes spawned by running applications
- scheduling applications
- memory management
- managing and optimizing hardware resources and dependencies
- managing and accessing I/O devices (keyboards, mice, disk drives, USB ports, network adapters and display,...)
- handling device and application system calls using various mechanisms such as hardware interrupts or device drivers

CPU modes to support kernel:

- *kernel mode*: code has unrestricted access to hardware; it is loaded in protected memory space and operates with highest privileges
- *user mode*: applications run with lower privileges; access to resources is made via system calls to kernel

Types of kernels:

- *microkernel*: delegates user services and processes in different address space; uses message-passing for communication; more flexibility and security (e.g. QNX - UNIX-based, real-time)
- *monolithic*: implements services in the same address space (e.g. most of UNIX-based kernels, Windows 9x)
- *hybrid*: tries to combine both (e.g. Windows 10, 11)

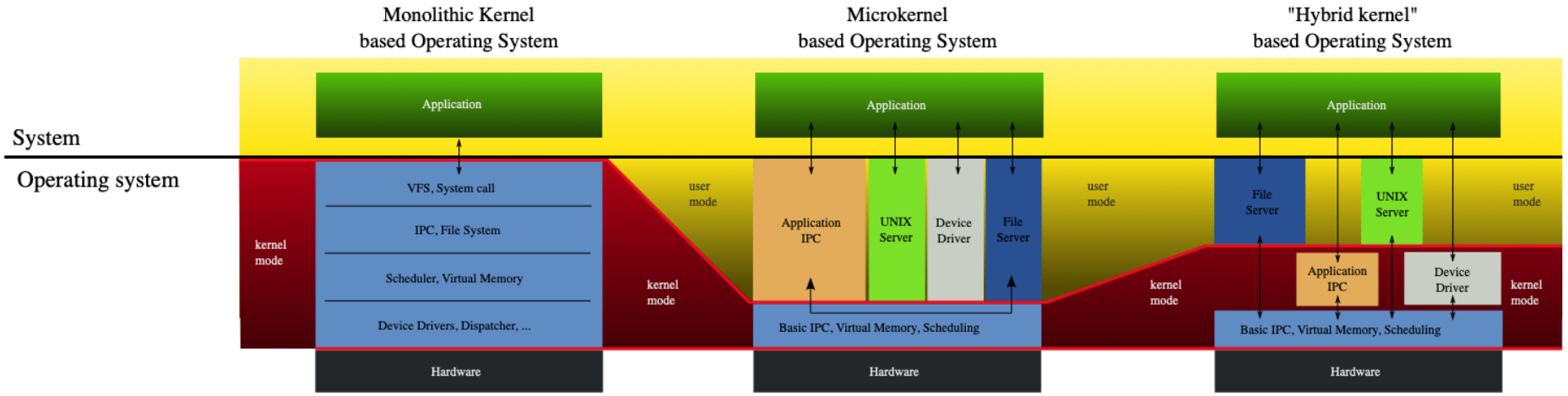
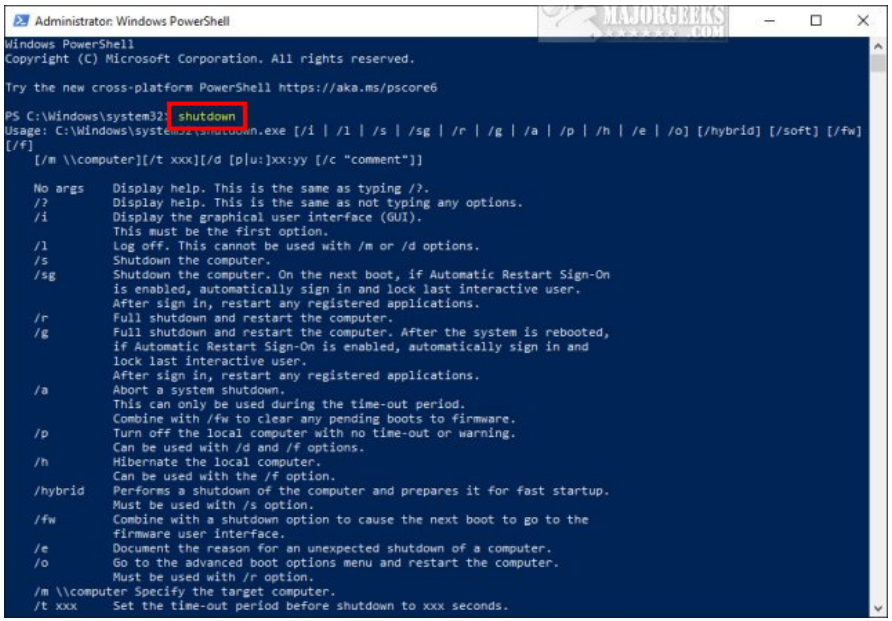


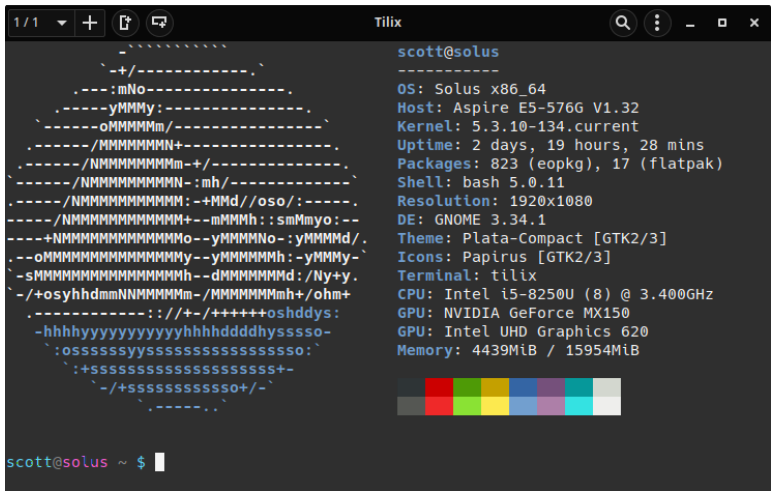
Figure: Comparison of three different kernel types (from Wikipedia)

Shells

- a program that allows users and programs to interact with OS services
- two modes: *command line interface (CLI)* and *graphical user interface (GUI)*
- CLI it has a specific *language* allowing the on line or scripted interaction with OS
- examples of CLI: Windows' Power Shell or UNIX's bash, tcsh, etc

Shells


Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Windows\system32> shutdown
Usage: C:\Windows\system32\shutdown.exe [/i | /l | /s | /sg | /r | /g | /a | /p | /h | /e | /o] [/hybrid] [/soft] [/fw] [/f]
[/m \\computer][/t xxx][/d [p|u:]xx:yy [/c "comment"]]
No args Display help. This is the same as typing /?.
/? Display help. This is the same as not typing any options.
/i Display the graphical user interface (GUI).
This must be the first option.
/l Log off. This cannot be used with /m or /d options.
/s Shutdown the computer.
/sg Shutdown the computer. On the next boot, if Automatic Restart Sign-On is enabled, automatically sign in and lock last interactive user.
After sign in, restart any registered applications.
/r Full shutdown and restart the computer.
/g Full shutdown and restart the computer. After the system is rebooted, if Automatic Restart Sign-On is enabled, automatically sign in and lock last interactive user.
After sign in, restart any registered applications.
/a Abort a system shutdown.
This can only be used during the time-out period.
Combine with /fw to clear any pending boots to firmware.
/p Turn off the local computer with no time-out or warning.
Can be used with /d and /f options.
/h Hibernate the local computer.
Can be used with the /f option.
/hybrid Performs a shutdown of the computer and prepares it for fast startup.
Must be used with /s option.
/fw Combine with a shutdown option to cause the next boot to go to the firmware user interface.
/e Document the reason for an unexpected shutdown of a computer.
/o Go to the advanced boot options menu and restart the computer.
Must be used with /r option.
/m \\computer Specify the target computer.
/t xxx Set the time-out period before shutdown to xxx seconds.


1 / 1
Tilix
scott@solus
OS: Solus x86_64
Host: Aspire E5-576G V1.32
Kernel: 5.3.10-134.current
Uptime: 2 days, 19 hours, 28 mins
Packages: 823 (eopkg), 17 (flatpak)
Shell: bash 5.0.11
Resolution: 1920x1080
DE: GNOME 3.34.1
Theme: Plata-Compact [GTK2/3]
Icons: Papirus [GTK2/3]
Terminal: tilix
CPU: Intel i5-8250U (8) @ 3.400GHz
GPU: NVIDIA GeForce MX150
GPU: Intel UHD Graphics 620
Memory: 4439MiB / 15954MiB
scott@solus ~ $

Questions?

Explorations

- go to `https://copy.sh/v86/` and instantiate some machines with Windows, DOS, and Linux and try to find your way around
- alternatively, try `https://www.pcjs.org/` for a more diverse selection of OSes (and other older programs)