OPERATING SYSTEMS

Operating-System Services, Design and Implementation

Venkatesh Prasad

Department of Computer Science

OPERATING SYSTEMS
Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the following resources and persons:

1. Slides of Operating System Concepts, Abraham Silberschatz, Peter Baer Galvin, Greg Gagne - 9th edition 2013 and some slides from 10th edition 2018
2. Some conceptual text and diagram from Operating Systems - Internals and Design Principles, William Stallings, 9th edition 2018
3. Some presentation transcripts from A. Frank – P. Weisberg
4. Some conceptual text from Operating Systems: Three Easy Pieces, Remzi Arpaci-Dusseau, Andrea Arpaci Dusseau

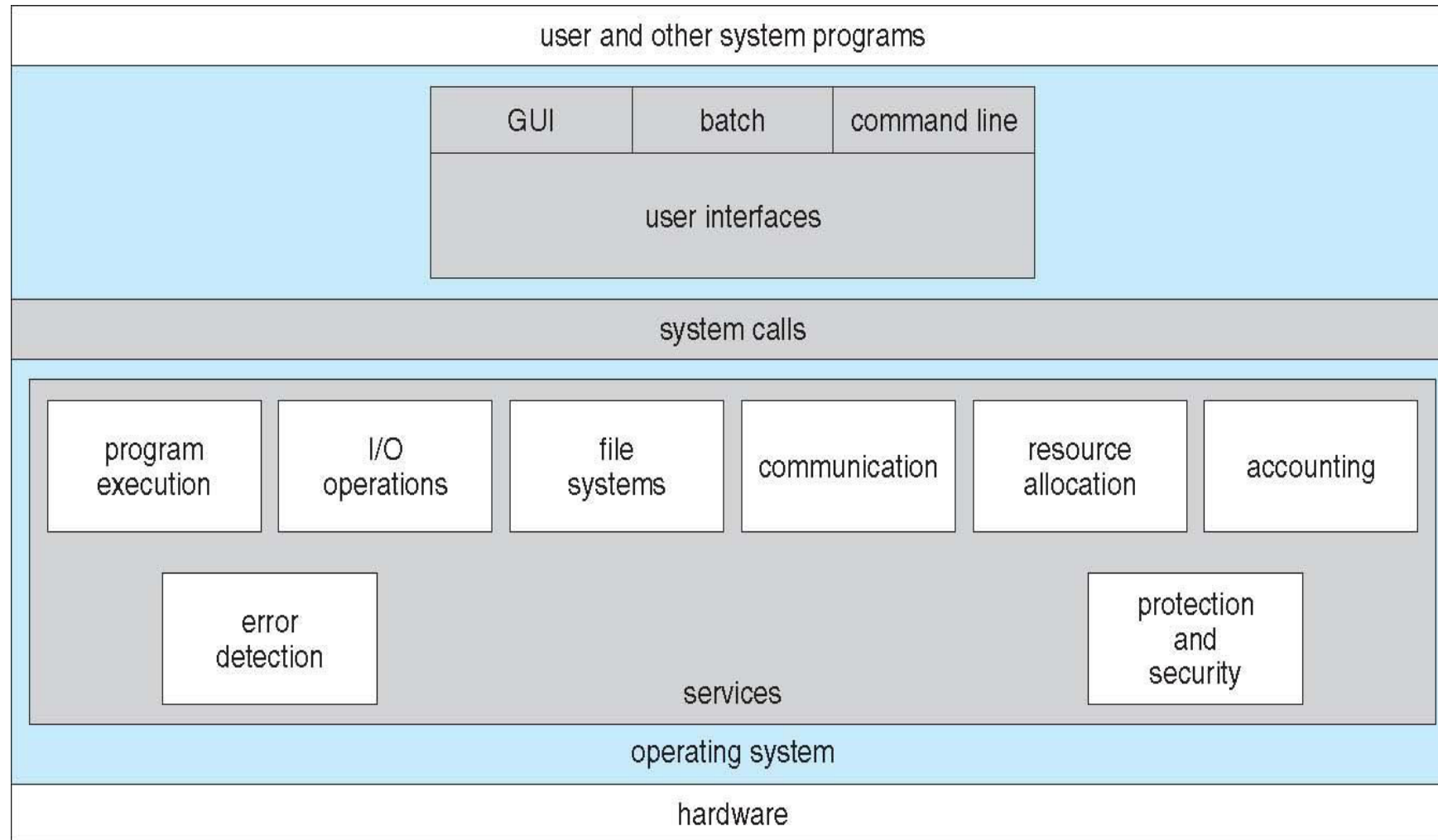# OPERATING SYSTEMS

Operating-System Services

Venkatesh Prasad

Department of Computer Science

Services

- Operating systems provide an environment for execution of programs and services to programs and users

- One set of operating-system services provides functions that are helpful to the user:

  - **User interface** - Almost all operating systems have a user interface (**UI**).

    - Varies between **Command-Line (CLI)**, **Graphics User Interface (GUI)**, **Batch**

  - **Program execution** - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)

  - **I/O operations** - A running program may require I/O, which may involve a file or an I/O device

A View of Operating System Services

■ One set of operating-system services provides functions that are helpful to the user:

- **File-system manipulation** - The file system is of particular interest. Programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.

- **Communications** – Processes may exchange information, on the same computer or between computers over a network

  ▸ Communications may be via shared memory or through message passing (packets moved by the OS)

Services

- **Error detection** – OS needs to be constantly aware of possible errors

  - May occur in the CPU and memory hardware, in I/O devices, in user program

  - For each type of error, OS should take the appropriate action to ensure correct and consistent computing

  - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

Services

- Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing

  - **Resource allocation -** When multiple users or multiple jobs running concurrently, resources must be allocated to each of them

    ▸ Many types of resources - CPU cycles, main memory, file storage, I/O devices.

  - **Accounting -** To keep track of which users use how much and what kinds of computer resources

Services

- **Protection and security -** The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other

  - ▸ **Protection** involves ensuring that all access to system resources is controlled

  - ▸ **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts

Operating-System Design and Implementation

- Design and Implementation of OS not "solvable" (i.e. no complete solutions to problems), but some approaches have proven successful
- Internal structure of different Operating Systems can vary widely
- Start the design by defining goals and specifications
- Affected by choice of hardware, type of system
- **User** goals and **System** goals
  - User goals – operating system should be convenient to use, easy to learn, reliable, safe, and fast
  - System goals – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient

Operating-System Design and Implementation

- Important principle to separate

  Policy: *What* will be done?
  Mechanism: *How* to do it?

- Mechanisms determine how to do something, policies decide what will be done

- The separation of policy from mechanism is a very important principle, it allows maximum flexibility if policy decisions are to be changed later (example – timer to prevent a user program from running too long)

- Specifying and designing an OS is highly creative task of **software engineering**

Implementation

- Much variation
  - Early OSes in assembly language
  - Then system programming languages like Algol, PL/1
  - Now C, C++
- Actually usually a mix of languages
  - Lowest levels in assembly
  - Main body in C
  - Systems programs in C, C++, scripting languages like PERL, Python, shell scripts
- More high-level language easier to **port** to other hardware
  - But slower
- **Emulation** can allow an OS to run on non-native hardware

THANK YOU

Venkatesh Prasad

Department of Computer Science Engineering

venkateshprasad@pes.edu