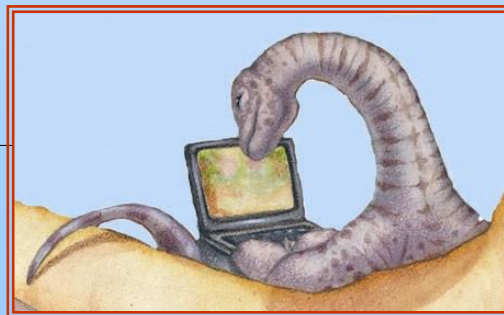


Chapter 1: Introduction





Objectives

- To provide a grand tour of the major operating systems components
- To provide coverage of basic computer system organization





What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware.
- Operating system goals:
 - Execute user programs and make solving user problems easier.
 - Make the computer system convenient to use.
- Use the computer hardware in an efficient manner.





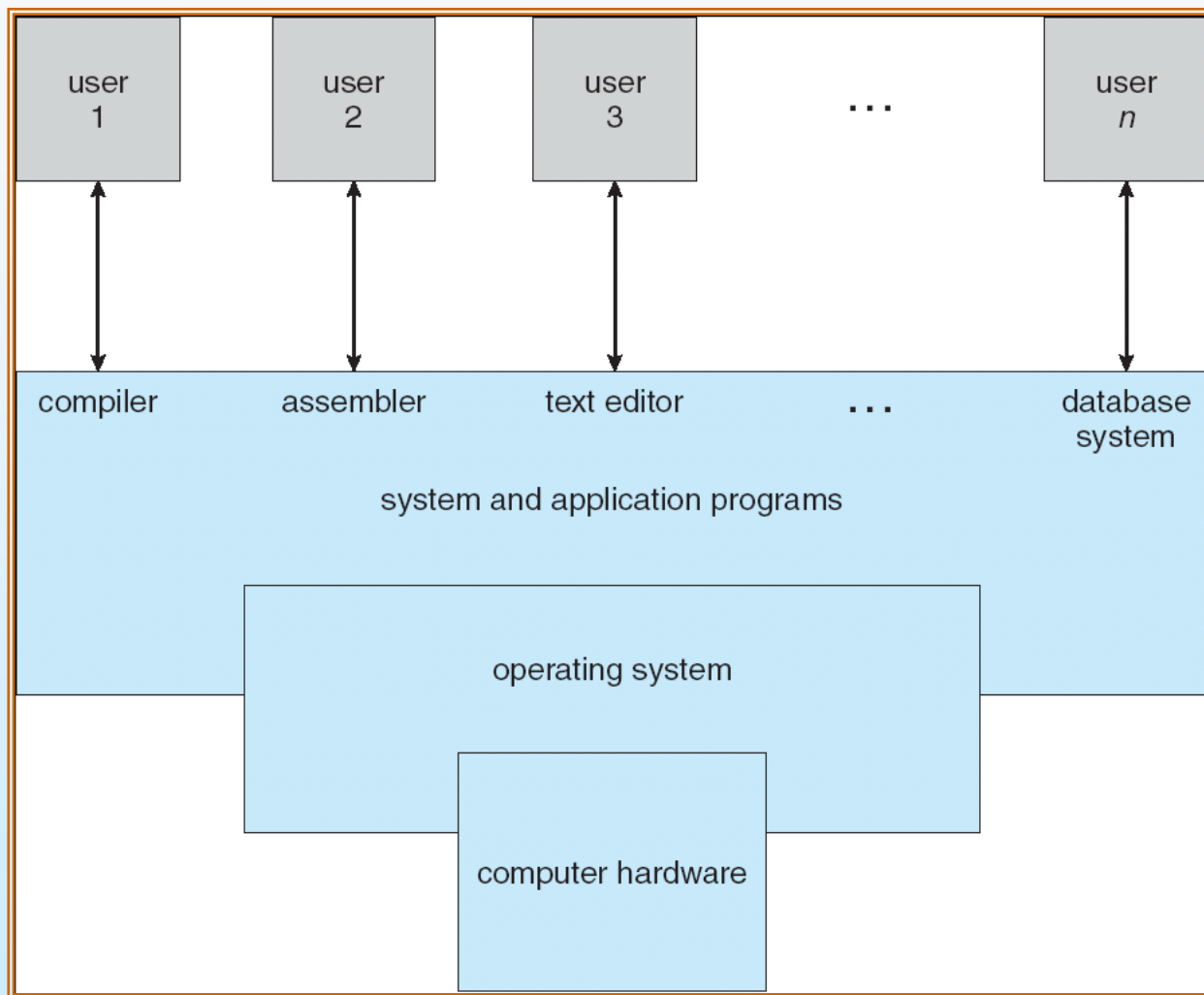
Computer System Structure

- Computer system can be divided into four components
 - Hardware – provides basic computing resources
 - ▶ CPU, memory, I/O devices
 - Operating system
 - ▶ Controls and coordinates use of hardware among various applications and users
 - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
 - ▶ Word processors, compilers, web browsers, database systems, video games
 - Users
 - ▶ People, machines, other computers





Four Components of a Computer System





Operating System Definition

- OS is a **resource allocator**
 - Manages all resources
 - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
 - Controls execution of programs to prevent errors and improper use of the computer





Operating System Definition (Cont.)

- No universally accepted definition
- “Everything a vendor ships when you order an operating system” is good approximation
 - But varies wildly
- “The one program running at all times on the computer” is the **kernel**. Everything else is either a system program (ships with the operating system) or an application program





Computer Startup

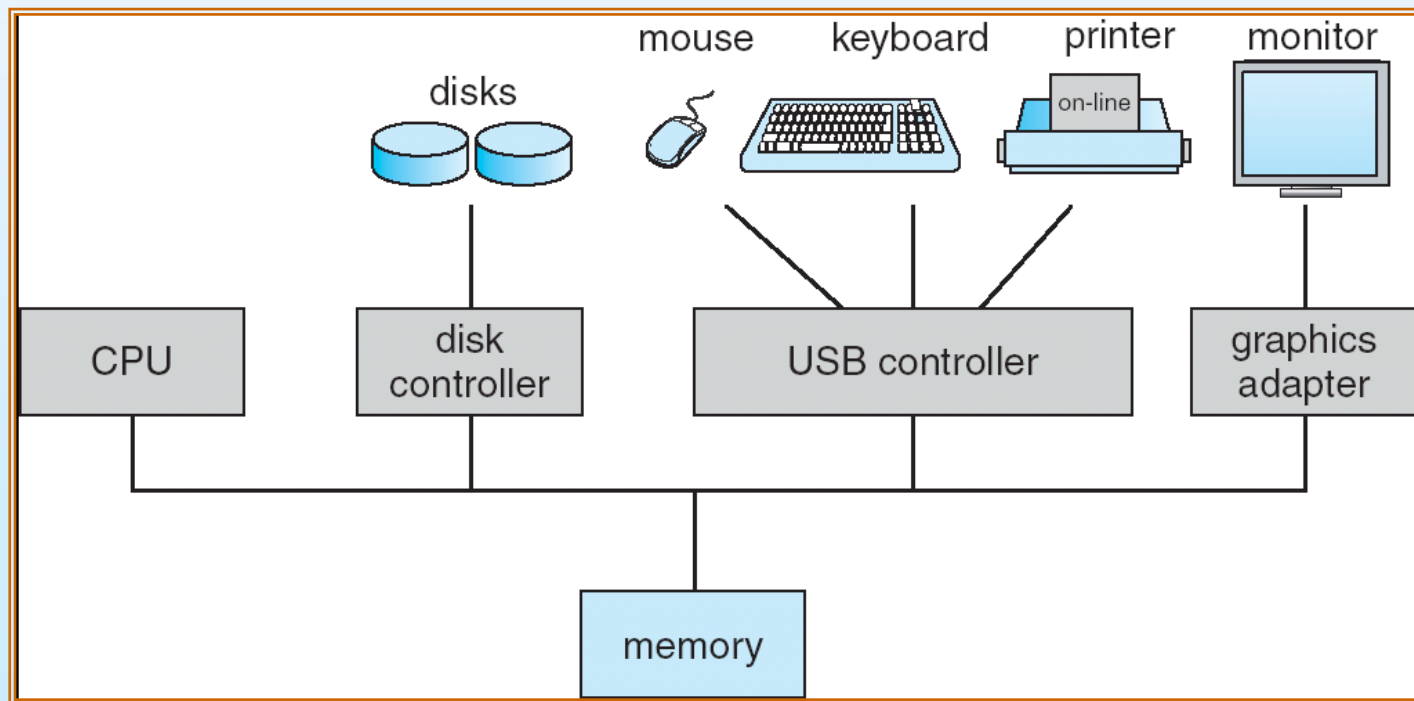
- **bootstrap program** is loaded at power-up or reboot
 - Typically stored in ROM or EEPROM, generally known as **firmware**
 - Initializes all aspects of system
 - Loads operating system kernel and starts execution





Computer System Organization

- Computer-system operation
 - One or more CPUs, device controllers connect through common bus providing access to shared memory
 - Concurrent execution of CPUs and devices competing for memory cycles





Computer-System Operation

- I/O devices and the CPU can execute concurrently.
- Each device controller is in charge of a particular device type.
- Each device controller has a local buffer.
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller.
- Device controller informs CPU that it has finished its operation by causing an *interrupt*.





Operating System (OS) services:

- An OS provides an environment for the execution of programs
- A set of OS services provides functions that are helpful to the user:
 - It provides a user Interface (comman-line or graphical)
 - It allows the execution of programs
 - It provides a means to do Input/Output (priviledge instruction)
 - It allows File System manipulation
- Another set of OS services ensures the efficient operation of the system itself:
 - Resource allocation
 - Accounting
 - Protection and security





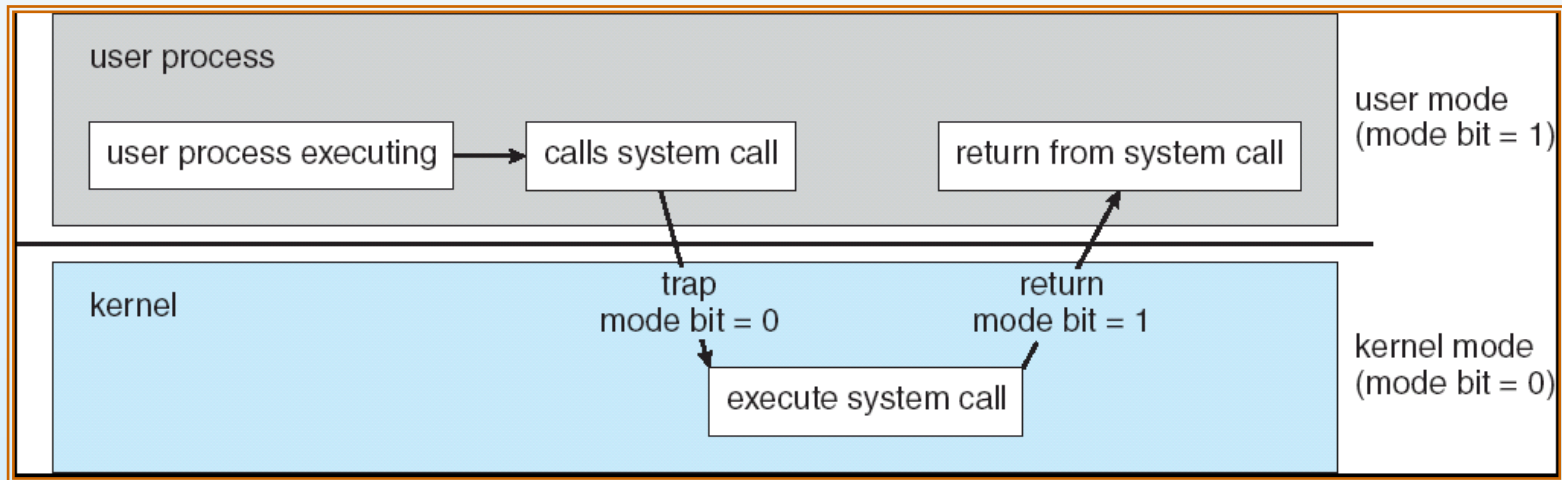
System Calls:

- **System calls provides an interface to the services made available by an OS**
- These calls are generally written in C or C++, although low-level tasks may need to be written in assembly language
- **Types of system calls:**
 - Process control: end, abort, load, execute, ...
 - File management: create file, delete file, open, close, ...
 - Device management: request device, release device, read, write, ...
 - Information maintenance: get/set time or date, ...
 - Communication: send, receive messages, ...





System Calls execution:





Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the *interrupt vector*, which contains the addresses of all the service routines.
- Interrupt architecture must save the address of the interrupted instruction.
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*.
- A *trap* is a software-generated interrupt caused either by an error or a user request.
- An operating system is *interrupt* driven.





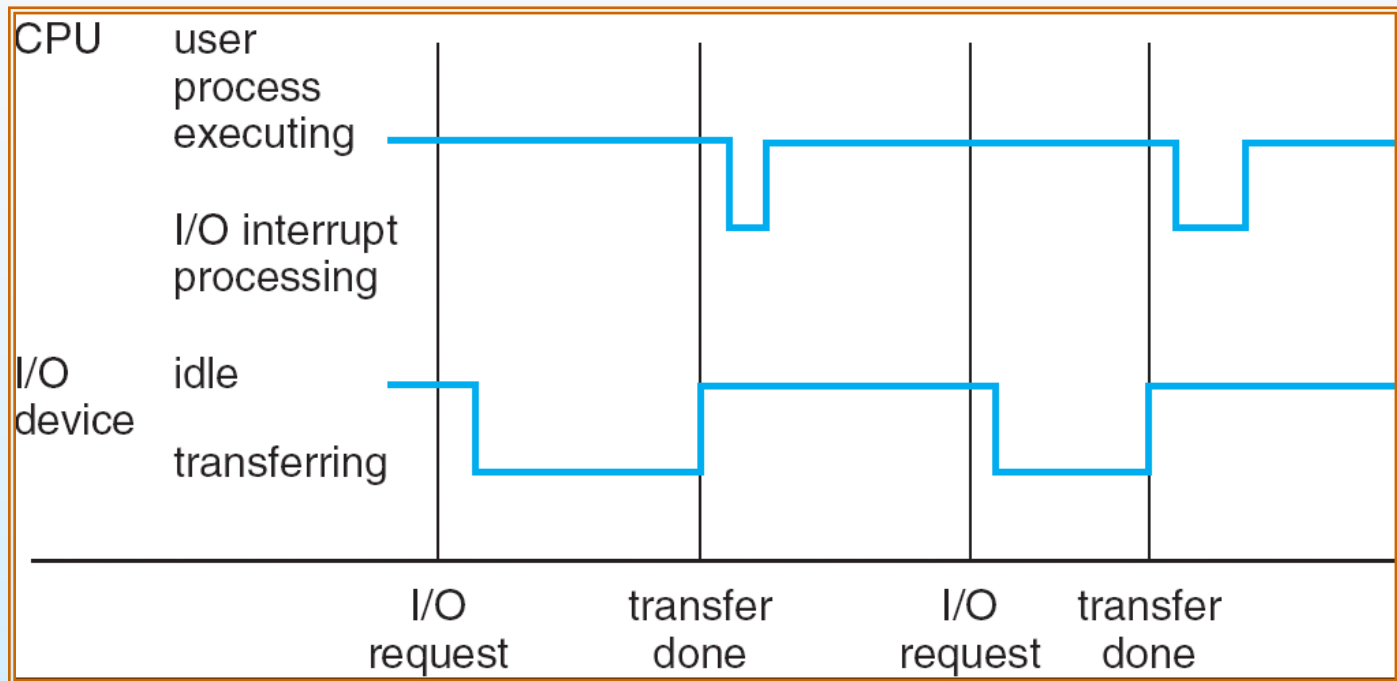
Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter.
- Determines which type of interrupt has occurred:
 - *polling*
 - *vectored* interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt





Interrupt Timeline





Operating-System Operations

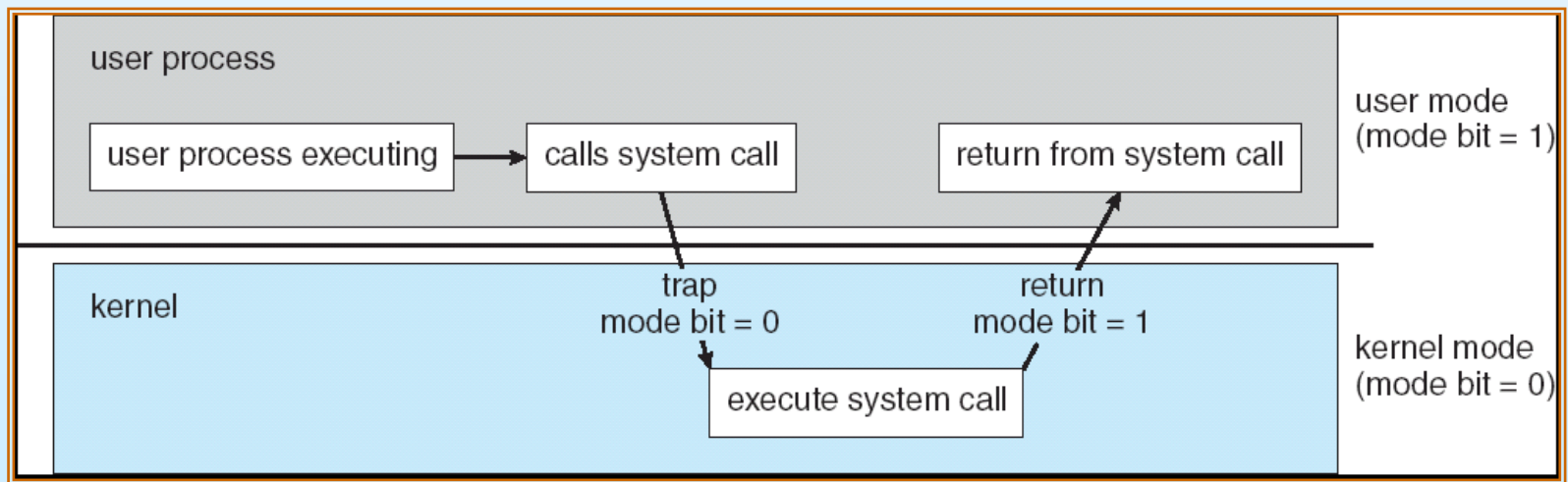
- Interrupt driven by hardware
- Software error or request creates **exception** or **trap**
 - Division by zero, request for operating system service
- Other process problems include infinite loop, processes modifying each other or the operating system
- **Dual-mode** operation allows OS to protect itself and other system components
 - **User mode** and **kernel mode**
 - **Mode bit** provided by hardware
 - ▶ Provides ability to distinguish when system is running user code or kernel code
 - ▶ Some instructions designated as **privileged**, only executable in kernel mode
 - ▶ System call changes mode to kernel, return from call resets it to user





Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
 - Set interrupt after specific period
 - Operating system decrements counter
 - When counter zero generate an interrupt
 - Set up before scheduling process to regain control or terminate program that exceeds allotted time



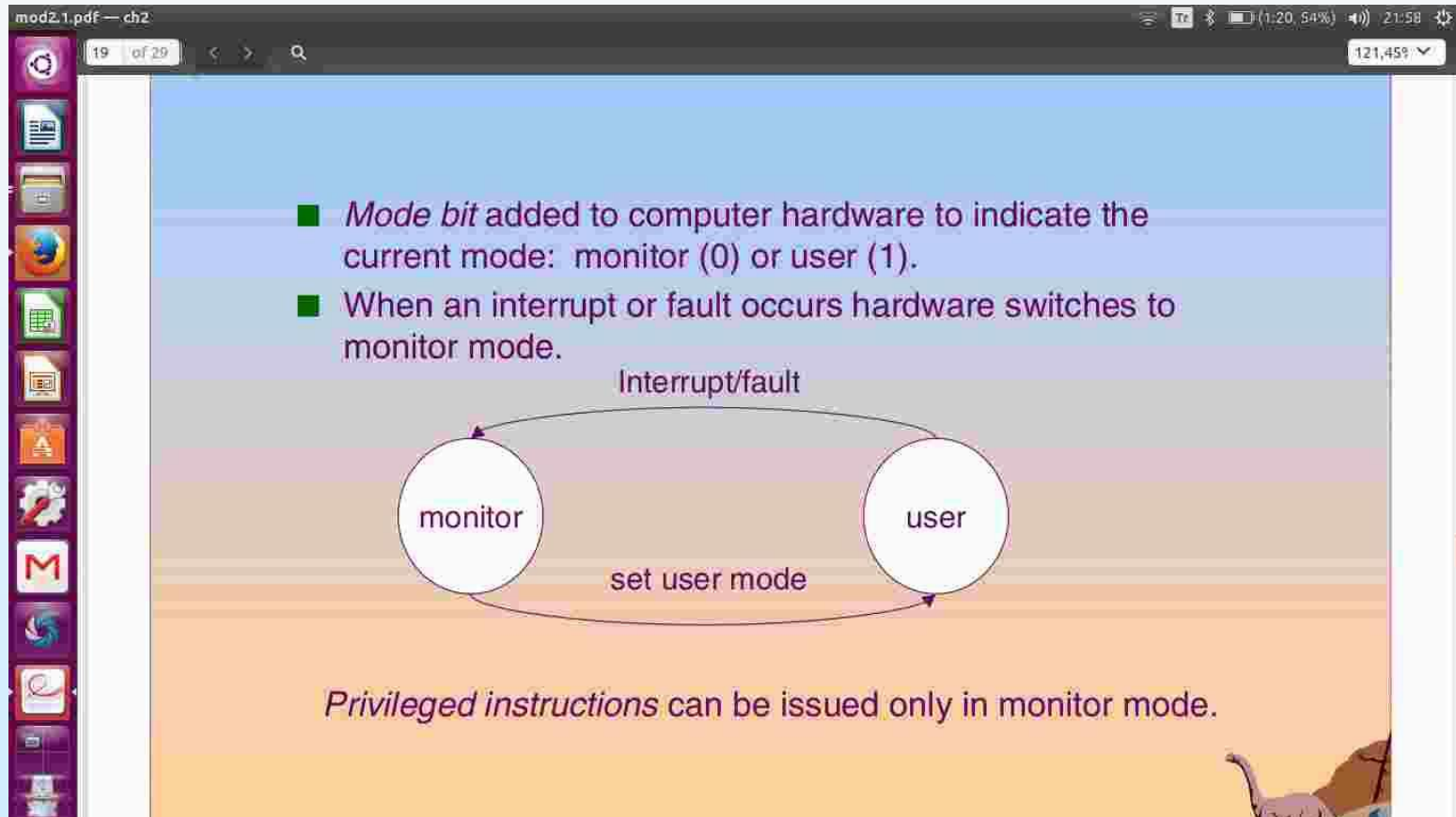


Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
 - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
 - User identities (**user IDs**, security IDs) include name and associated number, one per user
 - User ID then associated with all files, processes of that user to determine access control
 - Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file
 - **Privilege escalation** allows user to change to effective ID with more rights



Dual Mode (edition 6):





Input Output Protection (edition 6):

mod2.1.pdf — ch2

20 of 29

121,459

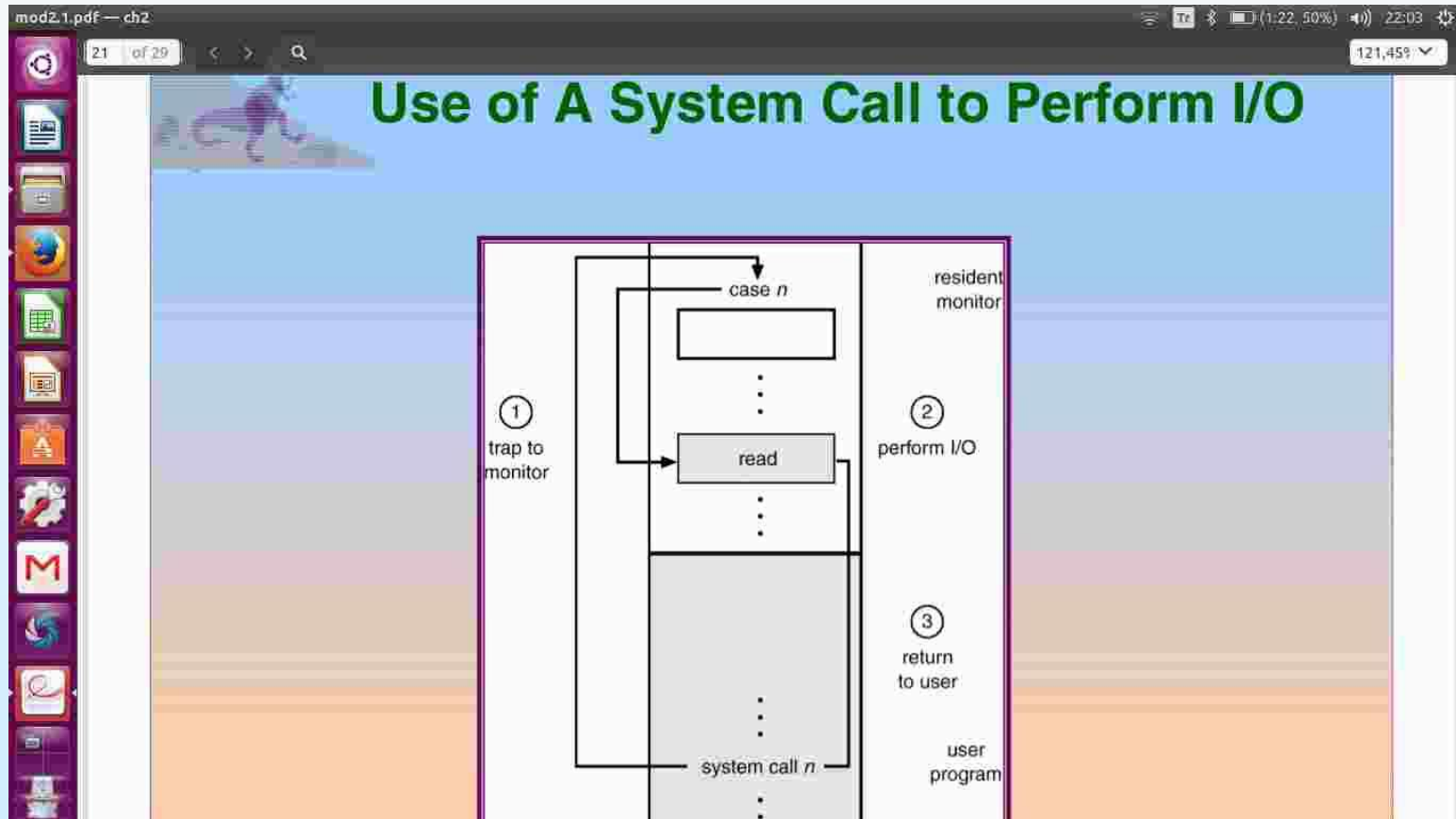
I/O Protection

- All I/O instructions are privileged instructions.
- Must ensure that a user program could never gain control of the computer in monitor mode (i.e., a user program that, as part of its execution, stores a new address in the interrupt vector).





System Call (edition 6):





Memory Protection (slide 1/3, edition 6):

mod2.1.pdf — ch2

22 of 29

121,45%

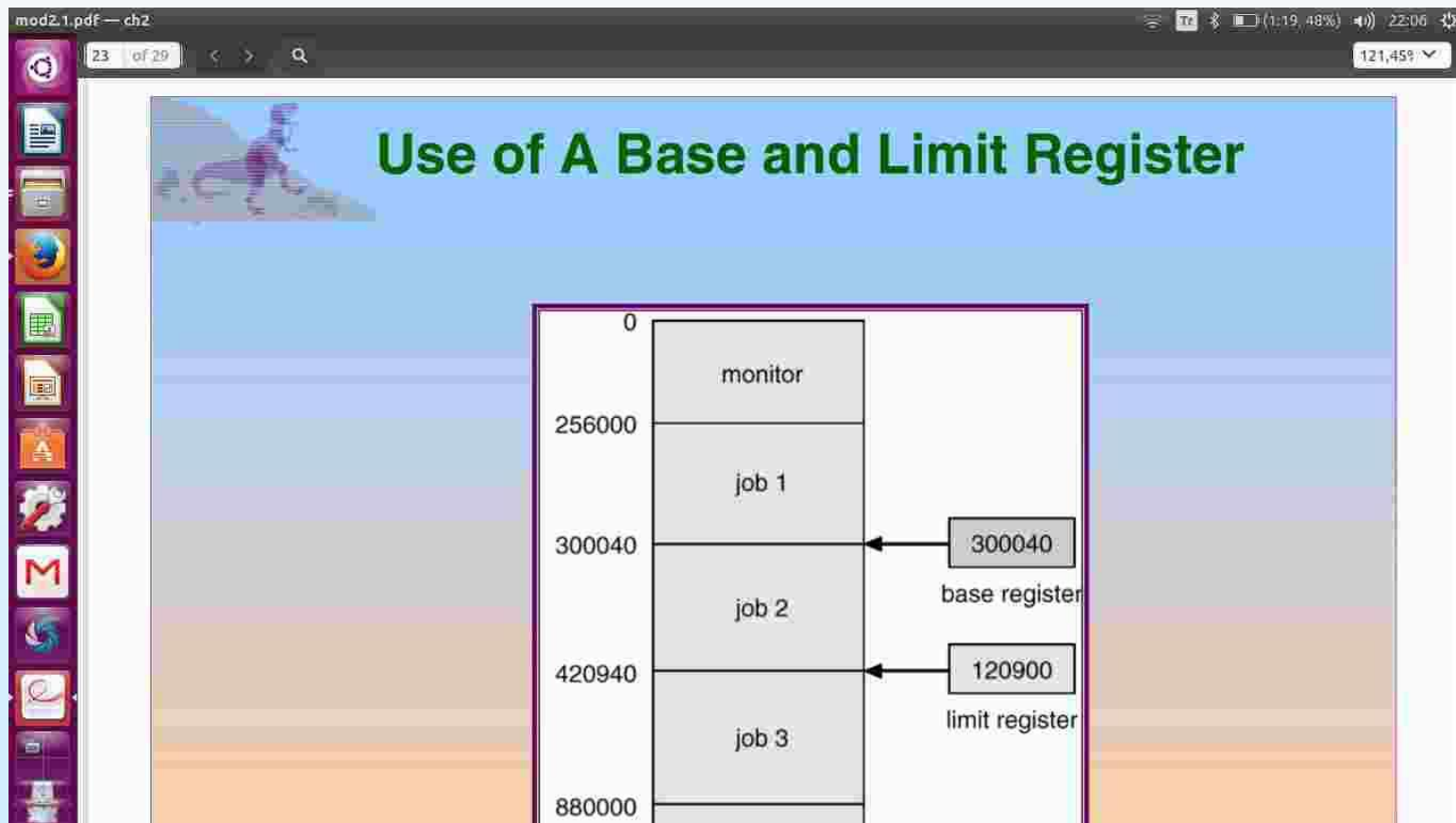
Memory Protection

- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:
 - ✦ **Base register** – holds the smallest legal physical memory address.
 - ✦ **Limit register** – contains the size of the range
- Memory outside the defined range is protected.



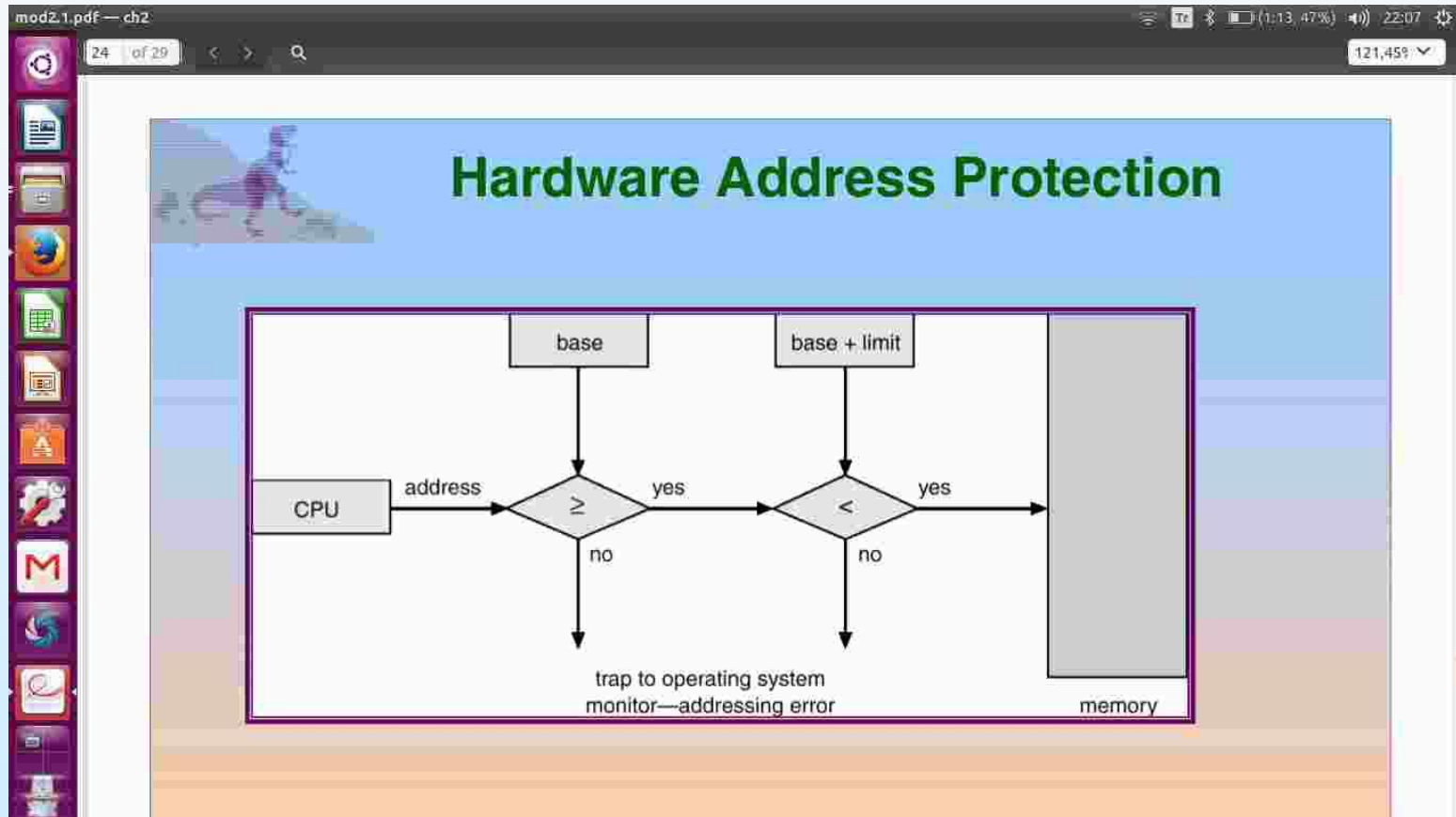


Memory Protection (slide 2/3, edition 6):





Memory Protection (slide 3/3, edition 6):





CPU Protection (edition 6):

mod2.1.pdf — ch2

26 of 29

121,45%

CPU Protection

- *Timer* – interrupts computer after specified period to ensure operating system maintains control.
 - ✦ Timer is decremented every clock tick.
 - ✦ When timer reaches the value 0, an interrupt occurs.
- Timer commonly used to implement time sharing.
- Time also used to compute the current time.
- Load-timer is a privileged instruction.

