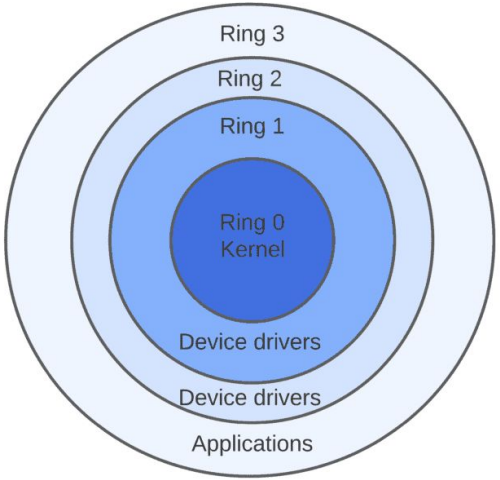
22 Aug 2025

Introduction & Background



Department of Computer Science and Engineering

International Institute of Information Technology, Bhubaneswar

Utkalika Satapathy

utkalika@iiit-bh.ac.in

Topics to be covered

01 - What happens on a function call ?

02 - Role of OS in running a process

03 - Concurrent execution & CPU virtualization

04 - Context Switching

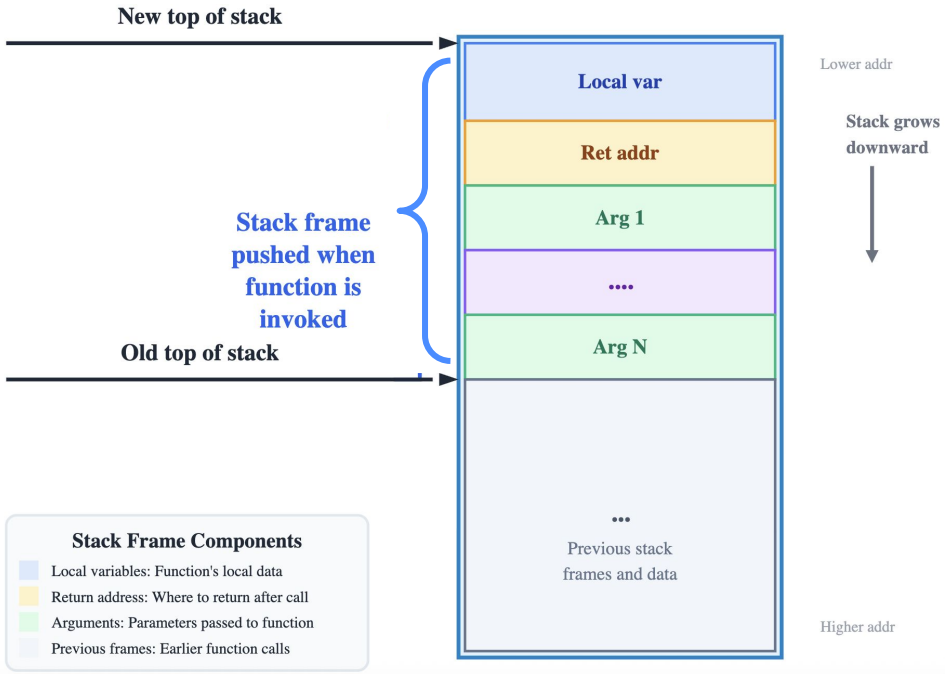
05 - User Mode and Kernel Mode

06 - System Calls

07 - Interrupts

What happens on a function call ?

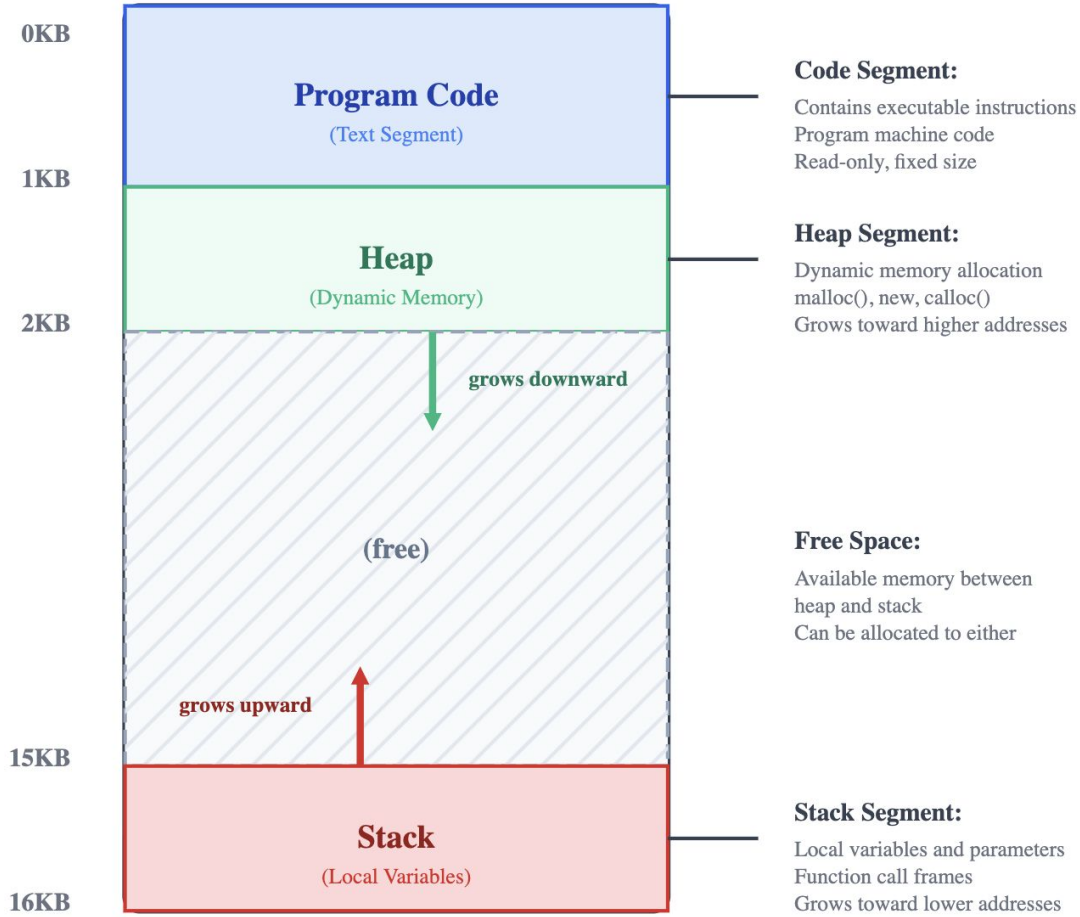
- Function arguments allocated on stack (in reverse order, by convention)
- Old PC (return addr) pushed on stack, PC jumps to function code
- Local variables allocated on stack
- Some register context saved too (more later)
- Now, new stack frame is ready on stack
- Function code runs using data on stack
- When function returns, all of the function memory is popped off the stack



Address space of a process

- OS gives every process the illusion that its memory image is laid out contiguously from memory address 0 onwards
  - This view of process memory is called the virtual address space
- In reality, processes are allocated free memory in small chunks all over RAM at some physical addresses, which the programmer is not aware of
  - Pointer addresses printed in a program are virtual addresses, not physical
- When a process accesses a virtual address, OS arranges to retrieve data from the actual physical address
- OS virtualizes memory for all processes, gives illusion of a virtual address space to processes

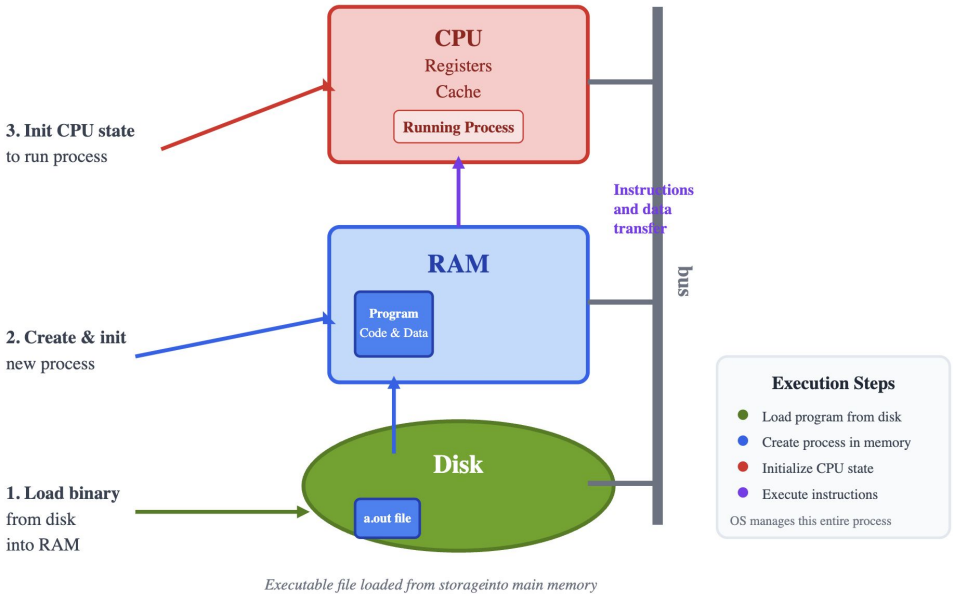
Process Memory Address Space Layout



- Code Segment: Fixed at program load time
- Heap: Managed by Malloc/Calloc and new/delete
- Stack: Automatically managed by functions
- Virtual Memory allows each process to have its own address space

Role of OS in running a process

- Allocates memory for new process in RAM
  - Loads code, data from disk executable
  - Allocates memory for stack, heap
- Initializes CPU context
  - PC points to first instruction
- Process starts to run
  - CPU runs user instructions now
  - OS is out of picture, but steps in later as needed

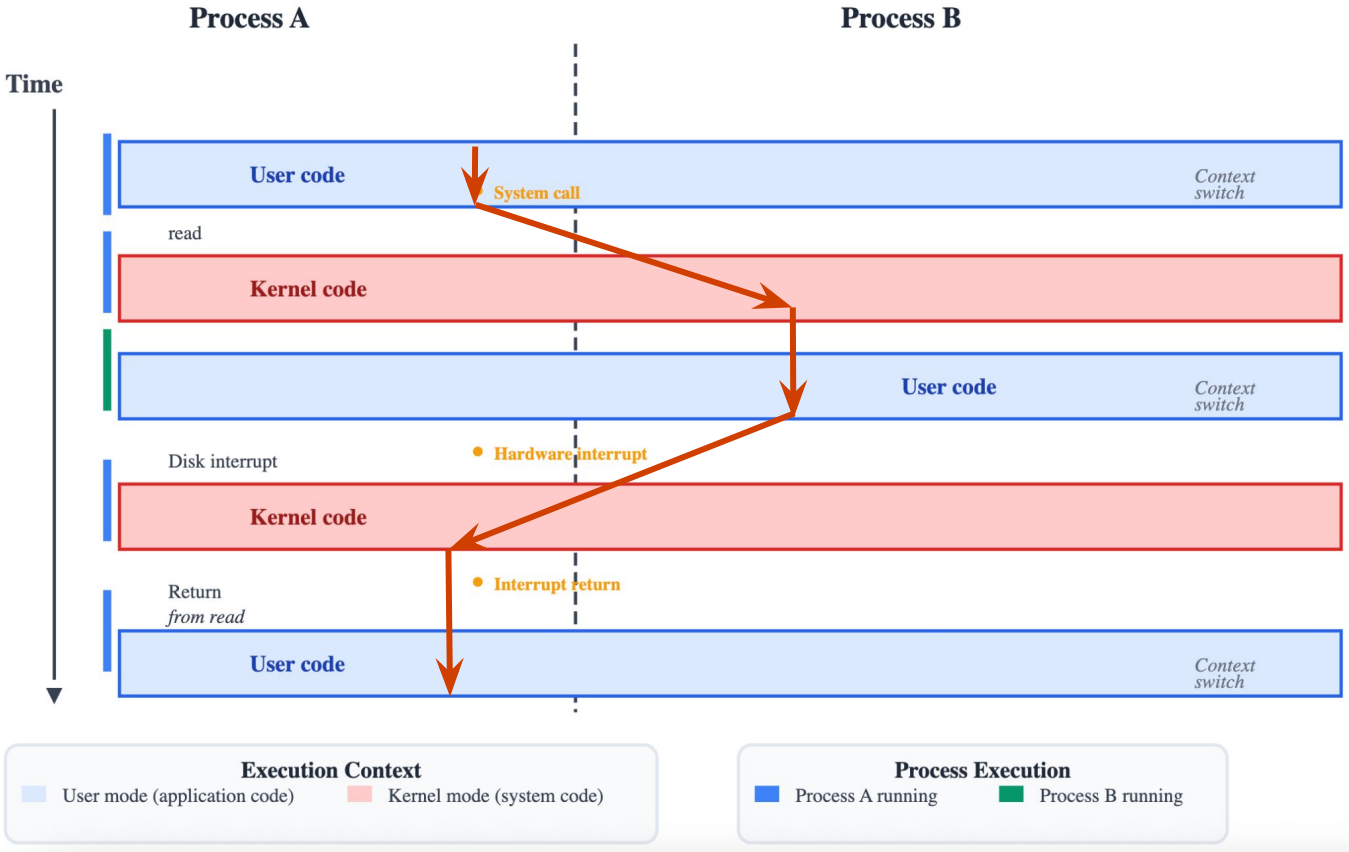


Executable file loaded from storageinto main memory

Concurrent execution & CPU virtualization

- CPU runs multiple programs concurrently
  - OS runs one process for a bit, then switches to another, switches again, …
- How does OS ensure correct concurrent execution?
  - Run user code of process A for some time
  - Pause A, save context of A, load context of B: context switching
  - Run user code of process B for some time
  - Pause B, save context of B, restore context of A, run A
- Every process thinks it is running alone on CPU
  - Saving and restoring context ensures process sees no disruption
- In this manner, OS virtualized CPU across multiple processes
- OS scheduler decides which process to run on which CPU at what time

Context Switching



Next Class We Will Talk About

- Process States
- Operations with examples from UNIX (fork, exec) and/or Windows.
- Process scheduling

Happy Learning !