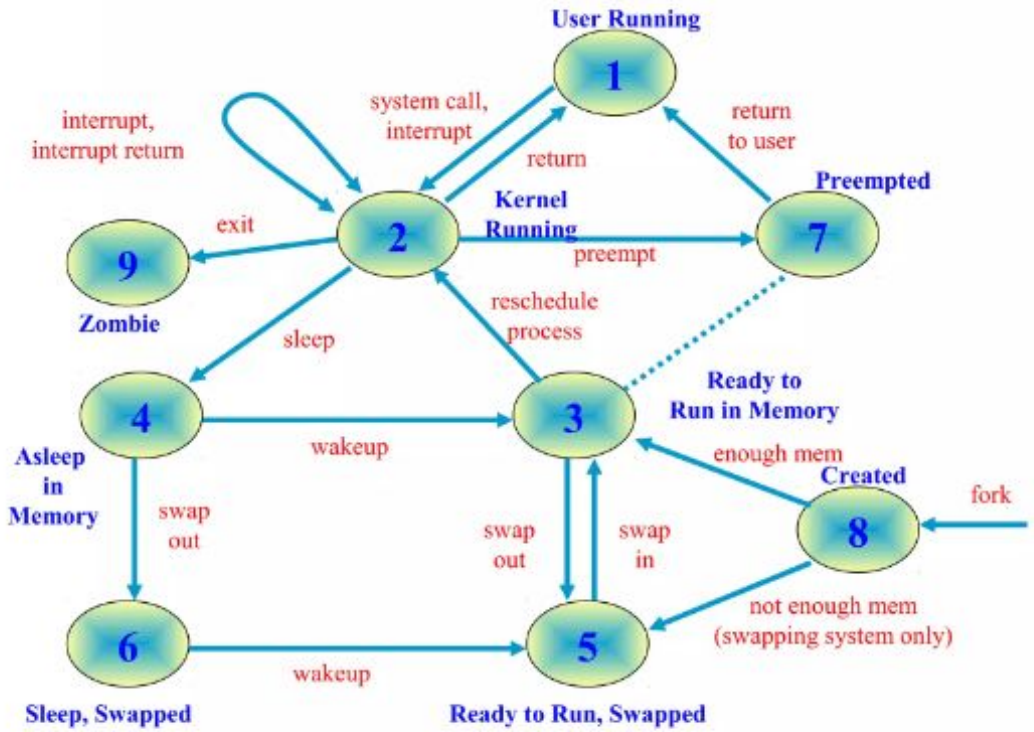
Structure of Processes

Contents

- Process States and Transitions
- Layout of System Memory
- The Context of a Process
- Saving the Context of a Process
- Manipulation of the Process Address Space

Process States and Transitions



Process States and Transitions

1. **User Running:** The process is executing in **user mode.**
2. **Kernel Running:** The process is executing in **kernel mode.**
3. **Ready to Run in Memory:** The process is not executing but is **ready to run** as soon as the kernel schedules it.
4. **Asleep in Memory:** The process is **sleeping** and resides in main memory.
5. **Ready to Run, Swapped:** The process is ready to run, but the **swapper (process 0) must swap the process into main memory** before the kernel can schedule it to execute.
6. **Sleep, Swapped:** The process is sleeping, and the **swapper has swapped the process to secondary storage** to make room for other processes in main memory.

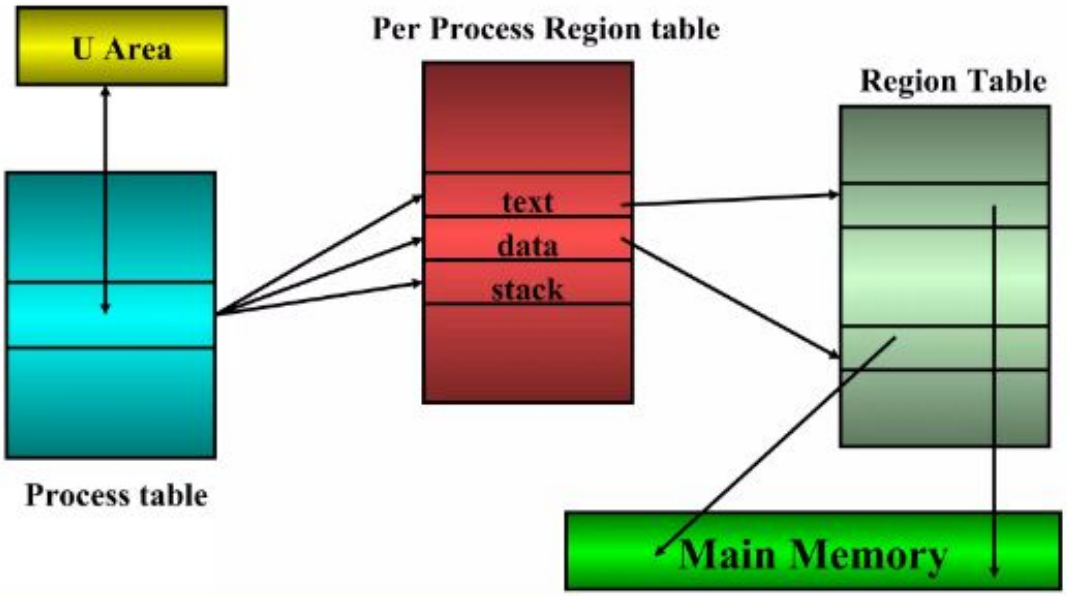
Process States and Transitions

7. **Preempted:** The process is returning from the kernel to user mode, but the **kernel preempts** it and does a context switch to schedule another process.
8. **Created:** The process is **newly created** and is in a transition state; the process exists, but it is not ready to run, nor is it sleeping. This state is the start state for all processes except process 0.
9. **Zombie:** The process executed the ***exit system call and is in the zombie state***. *The* process no longer exists, but it leaves a record containing an exit code and some timing statistics for its parent process to collect. The zombie state is the final state of a process.

State of a Process

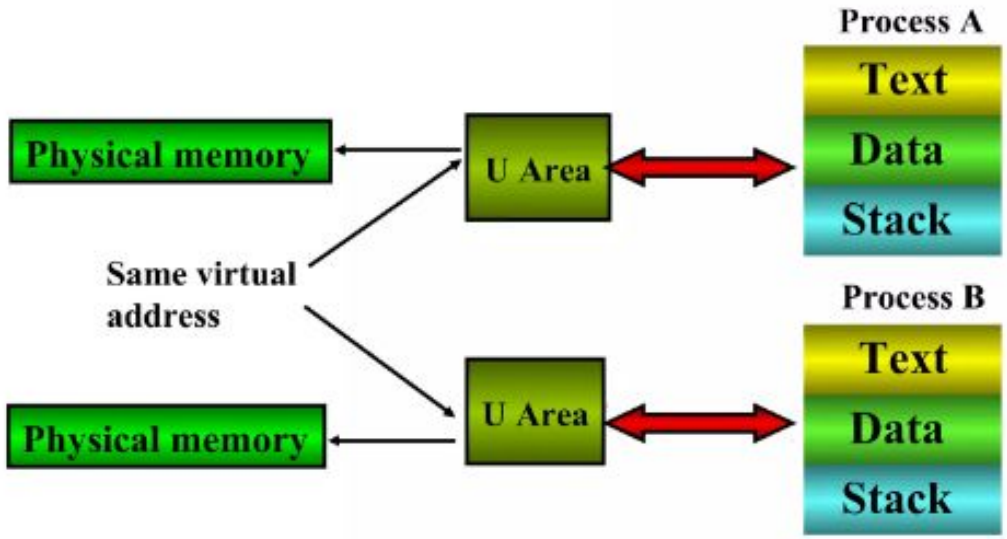
- **Process table entry**
  - Contains general fields of processes that must be always be accessible to the kernel.
- **U area**
  - Contains fields that only need to be accessible to the running process.

Data Structures for a Process



Per process region table allows independent processes to share regions.

User Area - U



- Each process has a user area.
- User area (U) has a fixed virtual address; it is mapped to different physical address.
- Each user area is mapped to a physical memory when process is loaded to memory.

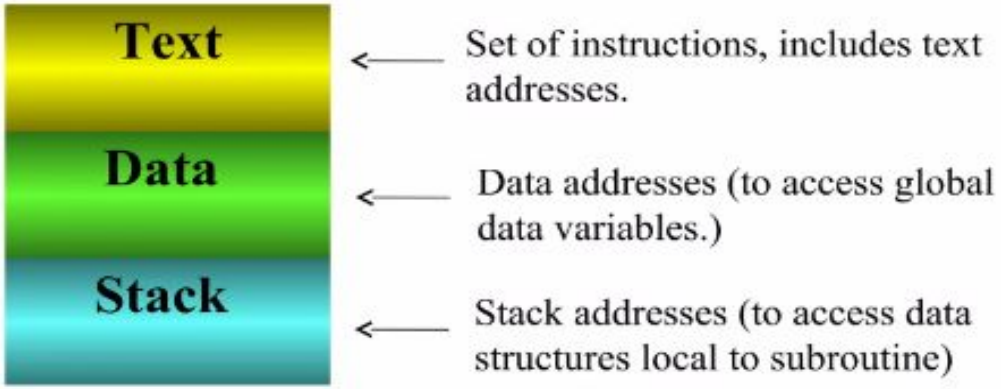
Process table fields

- **State field:** Identifies Process state. e.g. user running, kernel running, etc.
- **Fields** that allow the kernel to **locate the process and u area in Main Memory and Secondary Storage**. (Requires while context switch.)
  - **Process size:** kernel know how much space to allocate for the process.
- **User ID**
- **Process ID**
- **Event descriptor:**
  - Used when the process is in the "sleep" state. (Sleep/Wakeup).
- **Scheduling parameters:**
  - Allow the kernel to determine the order in which processes move to the states "kernel running" and "user running".
- **A Signal field:**
  - keeps the signals sent to a process but not yet handled.
- **Various timers:** process execution time, resource utilization, etc.

U Area fields

- **A pointer** to the process table entry.
- **Real and Effective User IDs**
- **Timer fields:**
  - Execution time in user mode
  - Execution Time in kernel mode
- **An error field:** keeps error during system call
- **Return value field:** result of system call.
- **I/O parameters:**
  - Amount of data transfer
  - Address of source and target etc.
- **The current directory and current root**
- **User file descriptor table**
- **Limit fields**
  - Restrict process size
  - Restrict size of the file it can write
- **The control terminal field:**
  - login terminal associated with the process, if one exists
- **An array**
  - indicates how the process wishes to react to signal.
- **A permission modes field.**

Layout of System Memory



•Process consists of 3 regions.
•Region is a contiguous area of the virtual address space of a process.

Regions

- Divides virtual address space of a process into logical regions.
- **Region :** Contiguous area of the virtual address space of a process

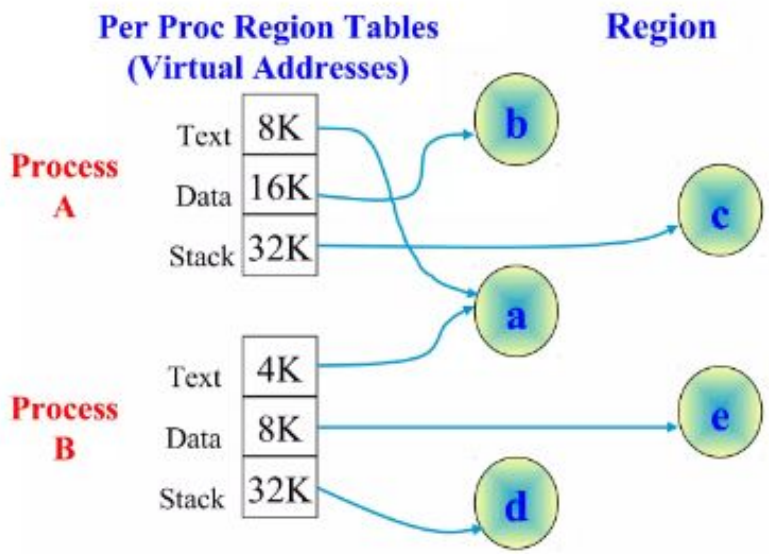
Per Process Region Table (Pregion)

- Pregion : region = File table : inode
- Each pregion entry points to the kernel region table
- Starting virtual (absolute) address of the region
- Permission filed:
  - read-only, read-write, read-execute

Kernel Region table

- Kernel region table contains the pointer to the page table which keeps the physical memory address

Regions



<Processes and Regions>

- A -> 8K
  B -> 4K
  reading,
  region 'a'
- Data and stack region for two processes are private.

Pages and Page Tables

- Memory Management polices are
  - ❖ Paging
  - ❖ Segmentation
- Pages and page table
  - ✓ Divides Physical Memory in to set of equal sized blocks called pages.
  - ✓ Page size range: 512 bytes to 4KB
  - ✓ Every memory location is addressed by a pair of

    (page number, byte offset in page)

Pages and Page Tables

Logical Page Number	Physical Page Number
0	177
1	54
2	209
3	17

Logical page numbers -> Physical page numbers

Virtual addresses of a region -> Physical machine addresses

Pages and Page Tables

- **Region** : A contiguous range of virtual addresses in a program.
  - **logical page number** : index into an array of physical page number
- Region table contains the **pointer to a table of physical page no.** called **page table**.
- Page table contain m/c dependent information such as permission bits(to allow reading & writing page).

Pages and Page Tables (contd.)

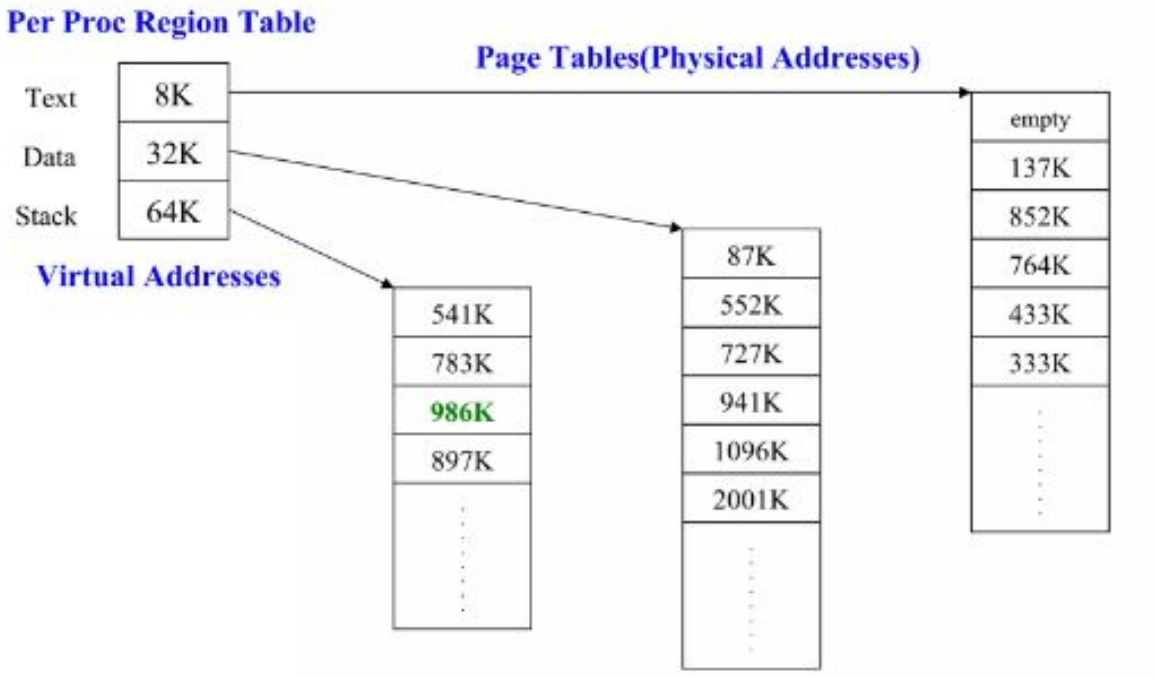


Fig: Mapping Virtual Addresses to Physical Address

Layout of Kernel

- Kernel executes in the context of a process
  - virtual memory mapping associated with the kernel is independent of all processes.
- Interrupt or system call
  - user mode -> kernel mode ; OS
  - user mode OS

Layout of Kernel

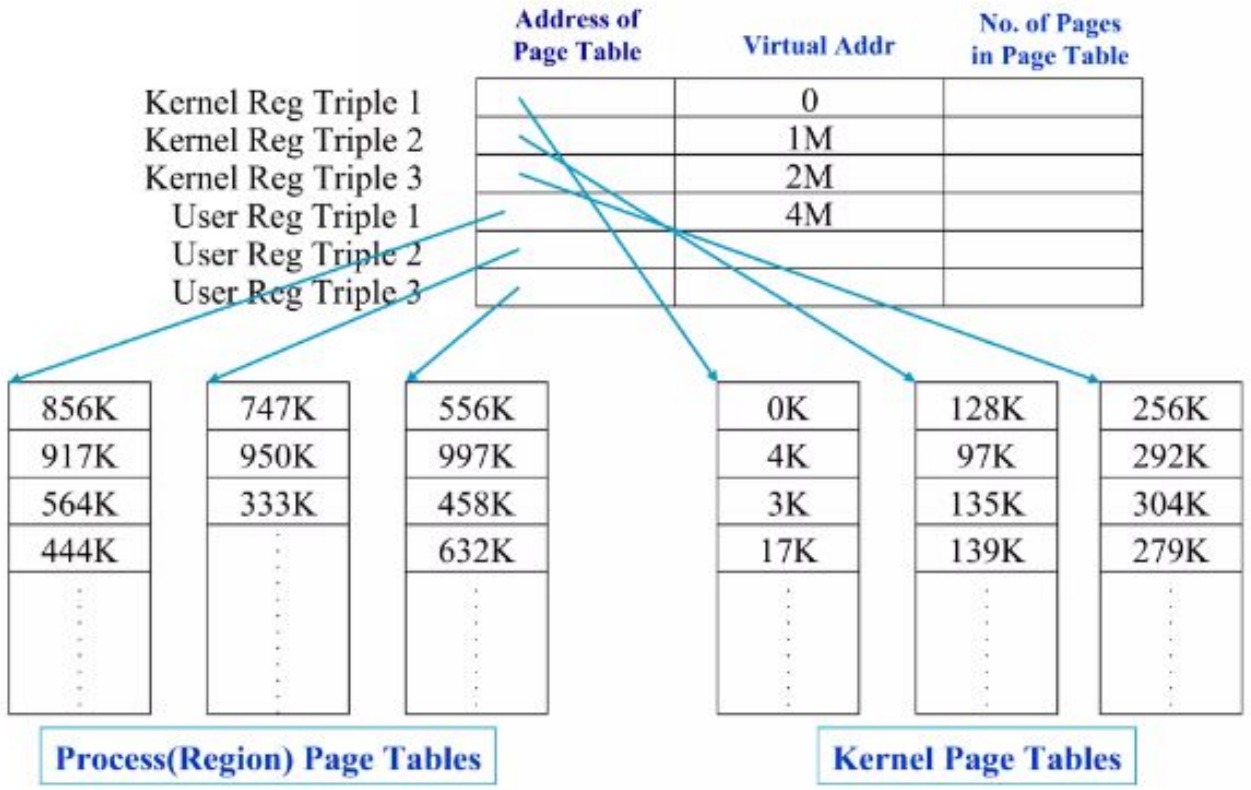


Fig: Changing Mode from User to Kernel

U-area (User)

- Every process has a private u area.
- The two physical address represent the *u areas* of two processes,
  - but the kernel accesses them via the same virtual address
- A process can access its *u area* when it executes in kernel mode.(not user mode)

U-area (User)

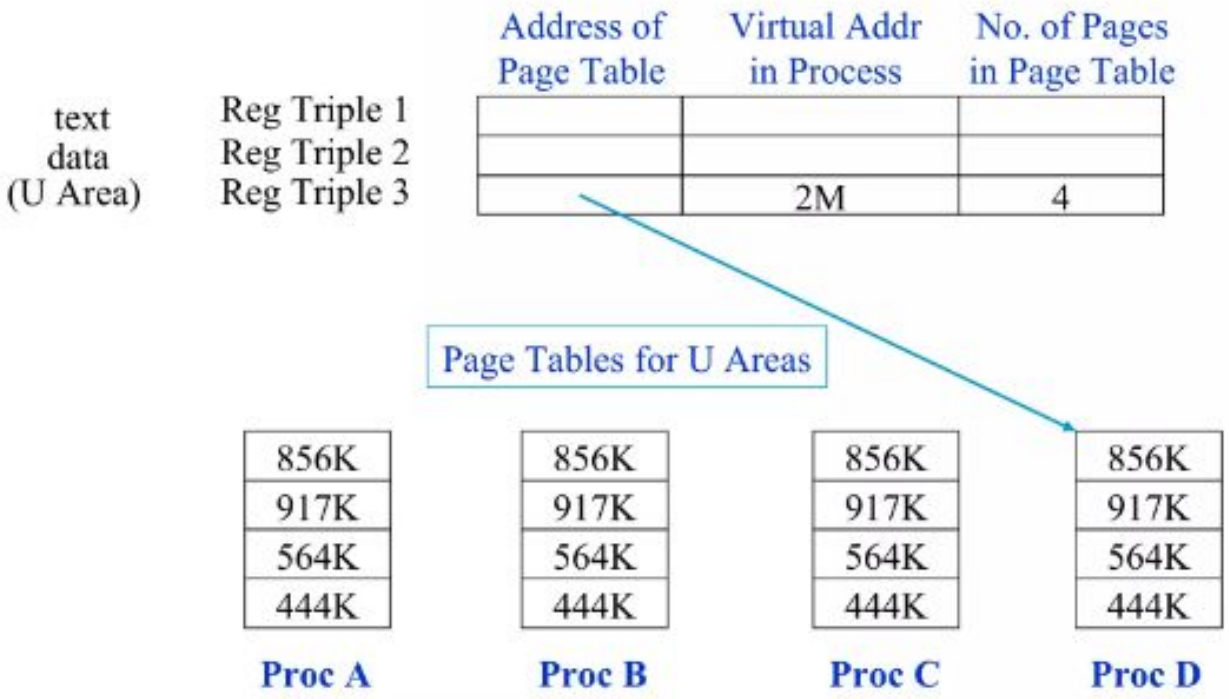


Fig: Memory Map of U Area in the Kernel