## Modes of Execution

| User Mode | System mode |
| --- | --- |
| Less Power | More Power |
| User Programs | Kernal of the OS |

## What to do to create new process?

- Process ID
- Space for Process
- Init PCB
- Setup linkages
- Creates/expand data structures

## Switching Processes

- What triggers a switch?
  - Time interrupt
  - I/O Interrupt
- Mode Switching vs Process Switching
  - User ⇌ Kernal Mode Switching
  - Not Process Switch
  - Operation Switch occurs

## Why would it Switch?

→ Interrupt: External to execution of current process

→ Trap: Access invalid memory during the execution
↳ Some error (!Div0)

→ Supervisor Call: when the process tries to do a kernal call
↳ Call to an operating system function

## Mode Switching

→ While switching; only the state info is saved & restored

→ More saved in process switch
↳ Heavy & Slow
User to Kernal

## Steps for Process Switch

① Save context
② Update current running PCB

③ Move process
④ Select another process
⑤ Update new PCB
⑥ Update new datastruct
⑦ Restore context of selected process

kernal to user

OS: Process? → Depends on the design

Can OS control itself? → Seperate kernal
- vs
- Everything is seperate

OS in Process
- vs
- within user, No Proc switch

Seperate OS Process
- Clean design; used in multiprocessor

↓

Microkernal architecture

Process Management in UNIX SVR4 → Swapper & Init are 0 & 1 ID processes.
- Parent of all the processes

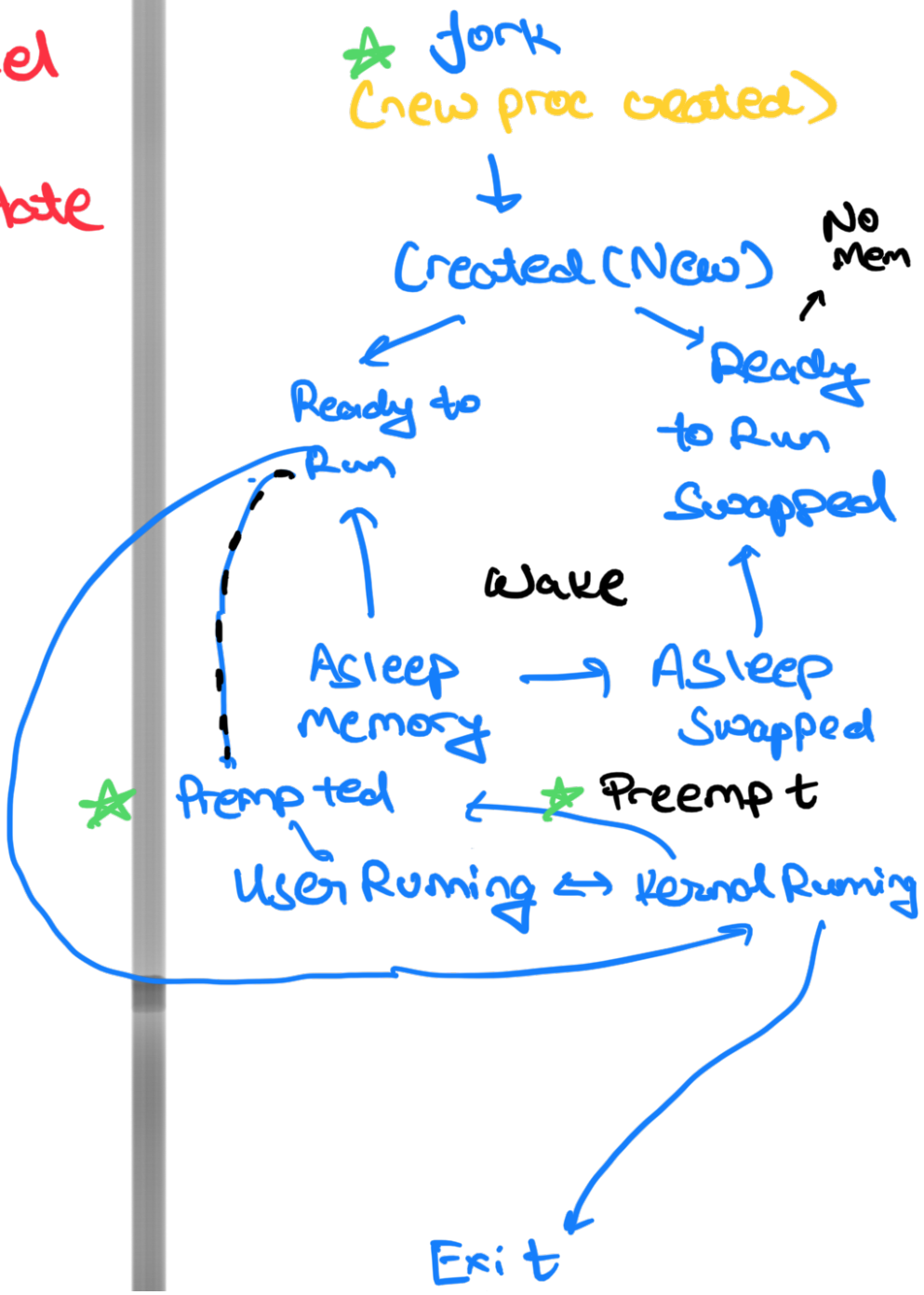
↓

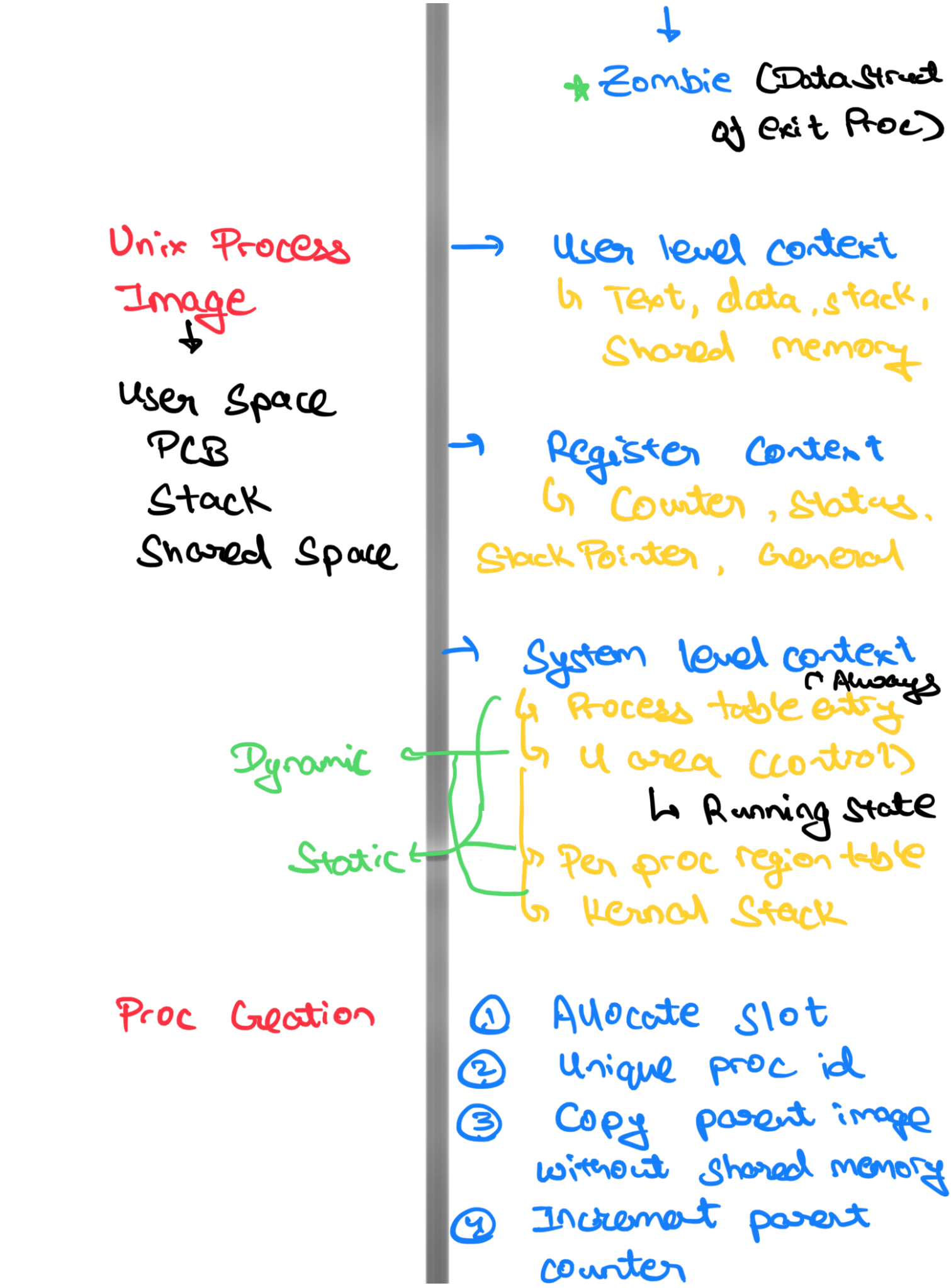
System V Release 4

→ System Proc:- Only Kernal Mode

→ User Proc:- User mode
↳ for user prog
Kernal mode for Kernal programs

7 state model
+
2 extra state




Zombie (Data stored of exit Proc)
Unix Process Image
User Space
PCB
Stack
Shared Space
User level context
Text, data, stack, Shared memory
Register Context
Counter, Status, Stack Pointer, General
System level context
Always
Process table entry
U area (control)
Running state
Per proc region table
Kernal Stack
Dynamic
Static
Proc Creation
1 Allocate slot
2 Unique proc id
3 Copy parent image without shared memory
4 Increment parent counter

⑤ Child → Ready to Run

⑥ Return child ID to Parent

**Summary:** Modes- Modes switching - Process Switching - Steps for process switch - OS control in 3 types - Diff in Kernel, Call as fn^n, or as the Microkernel - UNIX Case Study - System Model of 7+2 - fork, zombie, Preempt - User, Register, System context - U Area, Proc Table, PerProc Region, Kernel Stack; UK Dynamic - Process creation steps (child from Parent)