COP 4600

Objective # 3

So far.....

- **Obj#1 :** read "logon.dat" and created our event list.

- **Obj#2 :** function Boot() called. "boot.dat" was read. Kernel loaded and MEMMAP initialized.

main()

Interrupt_Handler()

- Saves state of interrupted program
- Gives control to Service Routine

(Simulator.c)

Interrupt() –

- get next event
- Update Simulation clock – 'CLOCK'

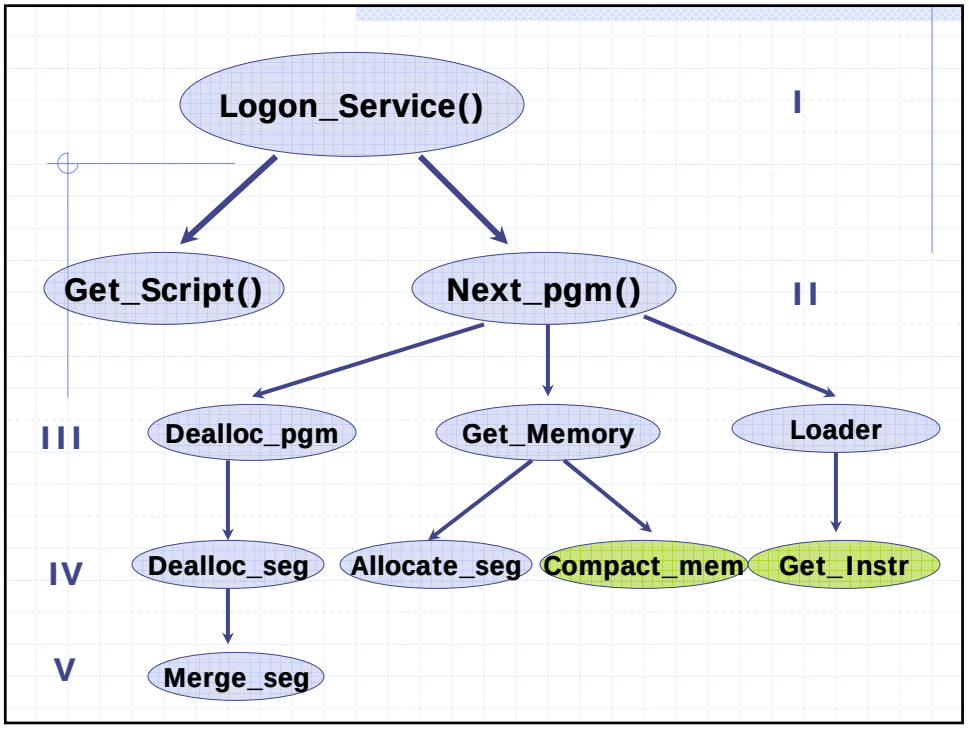
(OBJ # 1)

Objective # 3

Interrupt_Handler() → Logon_Service() →

→ Creates a PCB → Read "Script.dat" →

→ Allocate memory And Load Program from user's script .....



LEVEL - I

Logon_Service() :

[Allocates and Initializes the PCB for LOGON events]

- PCB stored in **"termtable"** (array of pointers to PCB's)
- Initialize PCB .
- Call **Get_Script (PCB)** // LEVEL II

  { Initialize the **process script** and **pgmid** }
- Call **Next_pgm (PCB)** // LEVEL II

  { Allocate and Load the next program from script }

LEVEL - II

Get_Script() :

[Reads a script from "script.dat"]

- Open **"script.dat"** file
- Read script till **LOGOFF** encountered
- Load script to **PCB -> script[.....]**

This gives us the list of programs a user will execute

LEVEL - II

Next_pgm() :

[Makes a transition to the next program in the script if possible]

- if (pcb->pgmid >= 0 && pcb->firstrb == NULL)
  Dealloc_pgm() // LEVEL III
- if (pcb->pgmid != NULL)
  return
- if (pcb->[++pcb->pgmid] == LOGOFF
  print to *.out
  return
- Get_memory() // LEVEL III { New Segment Table built }
- Loader() // LEVEL III { Load program into MEM }
- Assign Current PCB "R" Ready Status

LEVEL - III

Get_Memory() :

[Allocates Segment Table and sets up information for each segment]

Eg: editor.dat

- Open “editor.dat” and read the number of Segments
- Allocate a Segment Table
- Initialize each segment { acc bits , length etc as we did in **Boot()** }
- Allocate memory in **MEM** for each segments code.
  { This is done by calling
  Base = Alloc_seg(); // LEVEL IV
  if(base < 0)
  compact_mem(); // LEVEL IV – OBJ#6 }

LEVEL - III

Loader() :

[Reads editor.dat for instruction and loads each instruction into **MEM**]

- Call Get_Instr() {OBJ #2} to get opcodes and operand.
- Call Display_pgm() to echo to *.out

LEVEL - III

Dealloc_pgm() :

[Frees all allocated segments for the current program and then frees the segment table]

- **Call Dealloc_seg() for each segment in the pcb->segtable;**
- **free(pcb->segtable)**

LEVEL - IV

Dealloc_seg() :

[Return a segment to the free list]

Incase there are two adjacent segments in the free list .....call **Merge_Seg()**

LEVEL - IV

Alloc_seg() :

[Searches the **Free memory list** for free memory segments in MEM for a segment with a length equal to that requested]

- Return index in MEM incase enough memory found.
- Update Other Memory Parameters (eg: TotalFree)

LEVEL - V

Merge_seg() :

[Scans Memory list and if it finds two adjacent memory blocks it merges them into one]