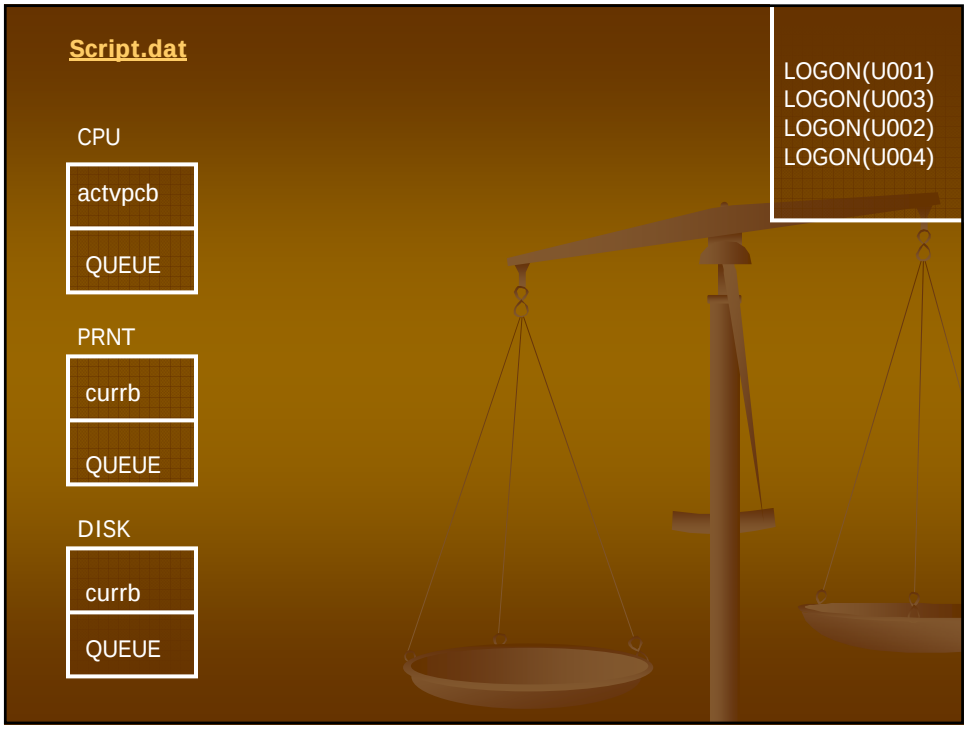
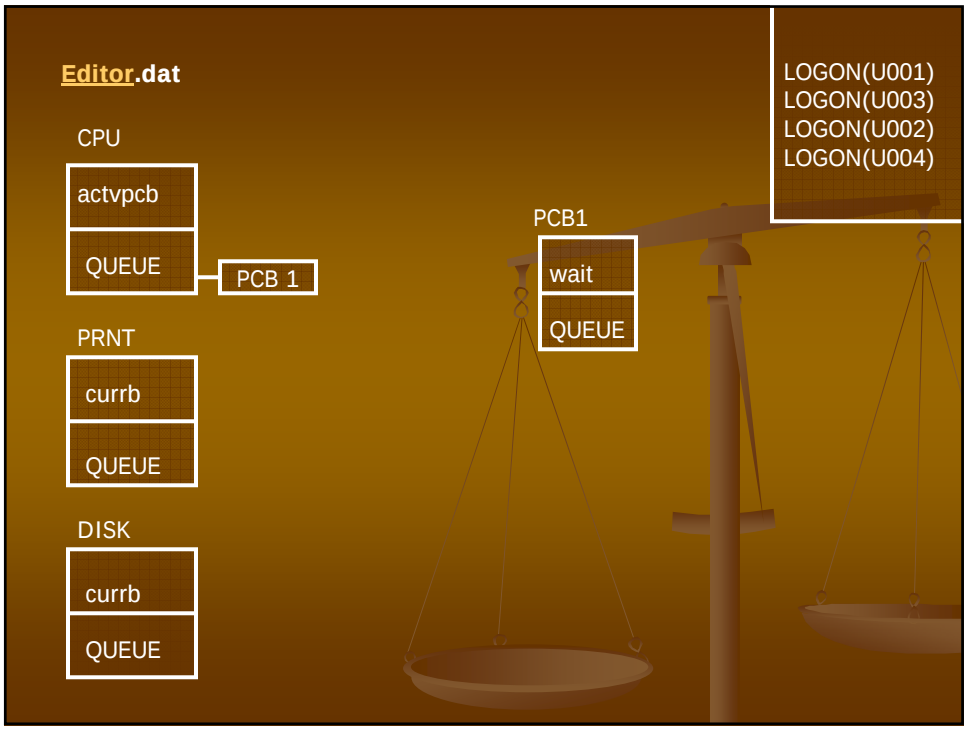
Summary of Routines

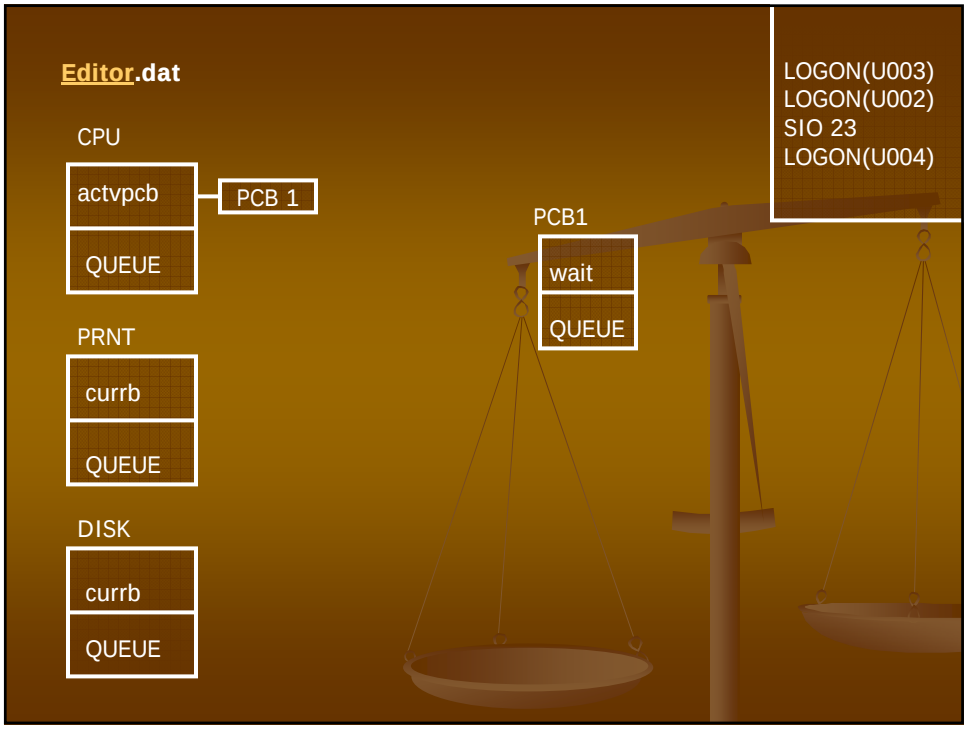
- **Add_cpuq()** - Adds a PCB to CPU Ready Queue.
- **Add_devq() -** Adds an **rb_type** node to the wait queue of a device.
- **Add_rblist() -** Adds an **rb_type** node to the RB-list of PCB.
- **Scheduler()** - Selects the Next Process from the CPU Ready Queue to give to the CPU for execution.
- **Dispatcher()** - Prepares scheduled PCB's program for execution and transfers it to CPU.

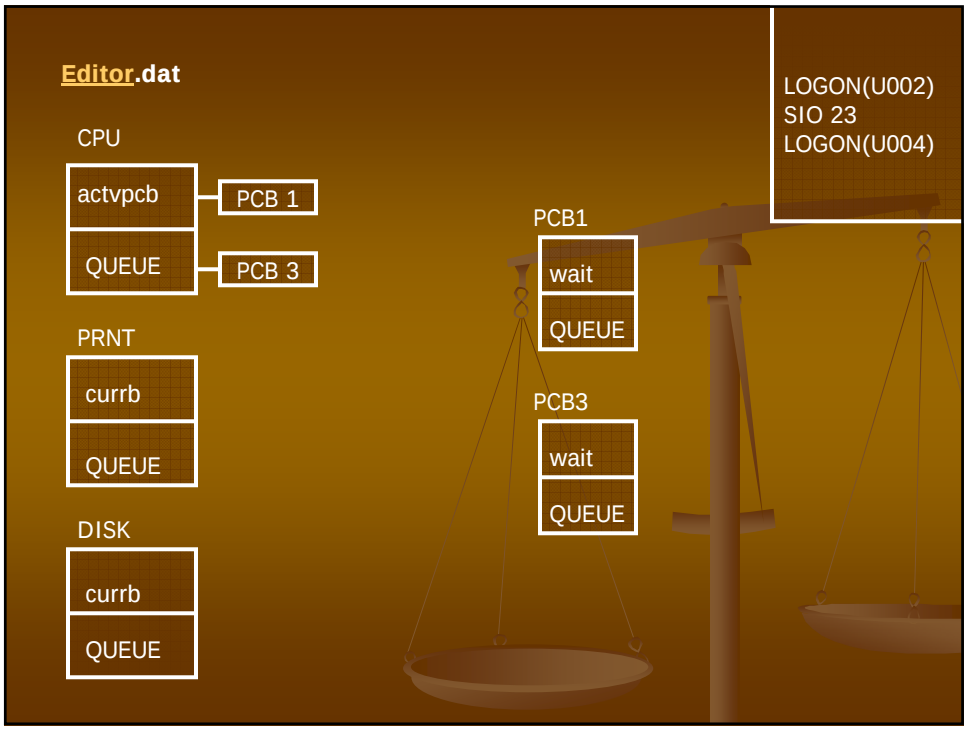
- **Sio_Service() -** Services interrupt to Start I/O (**NON-BLOCKING** function).Attempts to Start I/O on specified device if device not busy.
- **Wio_Service() -** Checks to see if a certain I/O operation that is waited upon is done.(**BLOCKING** function)**.**
- **Eio_Service() - Services** Interrupts from I/O Devices indicating the End of a previously assigned I/O operation. Attempts to Start I/O on the next node in Devices Queue.

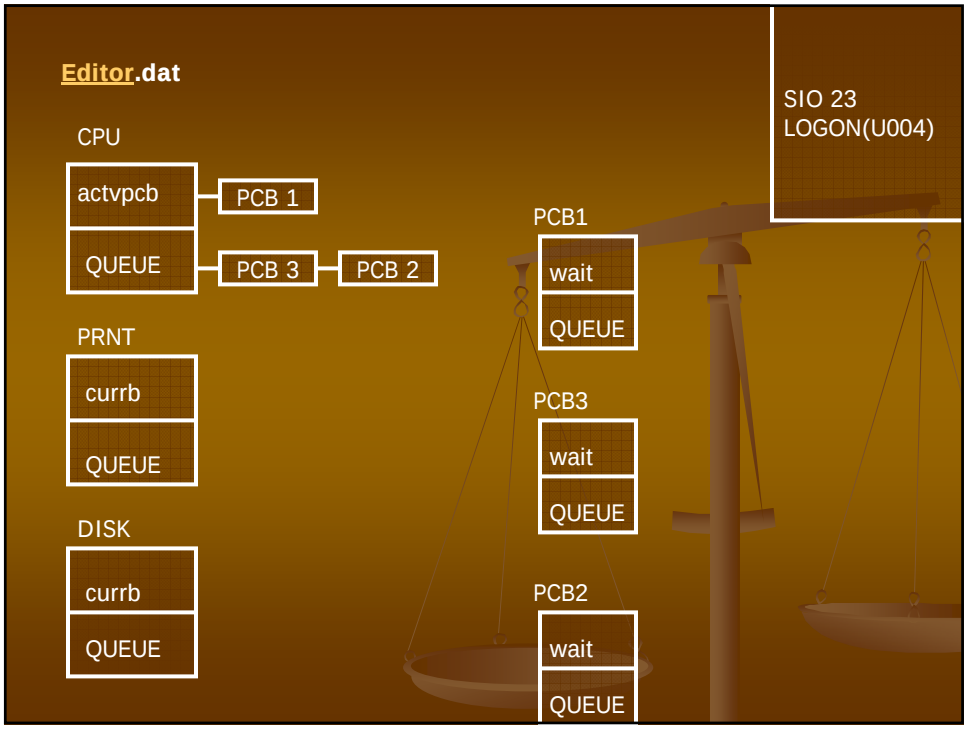
- **Start_IO() -** Simulates Device providing service to nodes waiting in its Queue.
- **Find_rb() -** Searches PCB's RB-List for 'rb'.
- **Delete_rb() -** Deallocates 'rb' node if found in PCB's RB- List.
- **Purge_rb() -** Purges all RB's with Status as 'D' – Done.
- **Load_Map() -** Initializes MEMMAP h/w with Dispatched Programs Segment Table

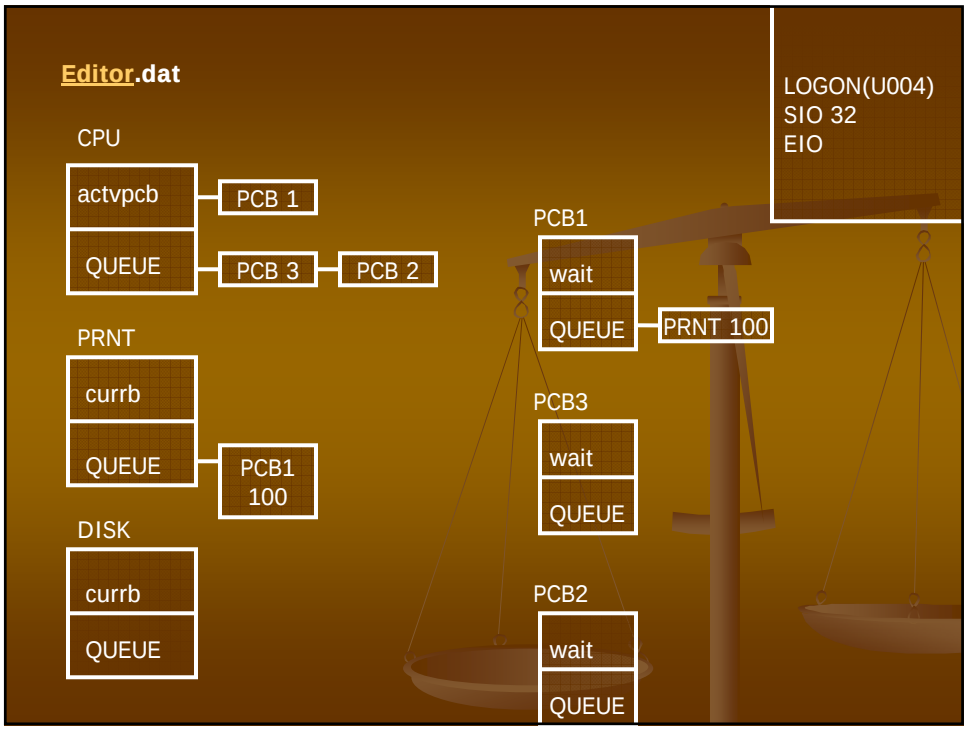

Script.dat
CPU
actvpcb
QUEUE
PRNT
currb
QUEUE
DISK
currb
QUEUE
LOGON(U001)
LOGON(U003)
LOGON(U002)
LOGON(U004)

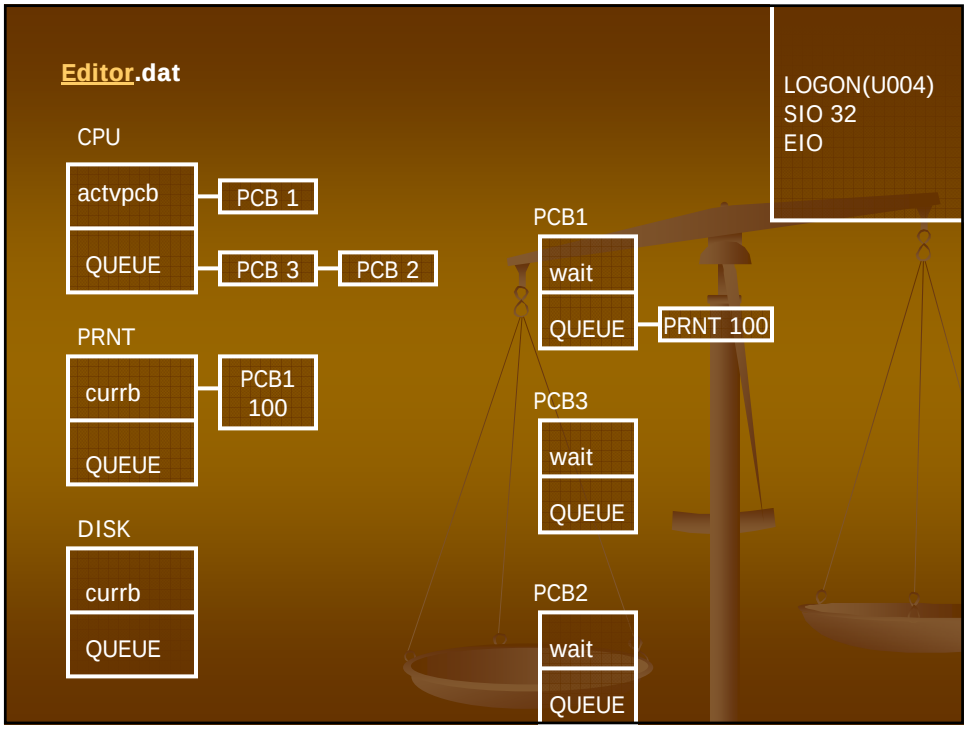

Editor.dat
CPU
actvpcb
QUEUE
PCB 1
PRNT
currb
QUEUE
DISK
currb
QUEUE
PCB1
wait
QUEUE
LOGON(U001)
LOGON(U003)
LOGON(U002)
LOGON(U004)

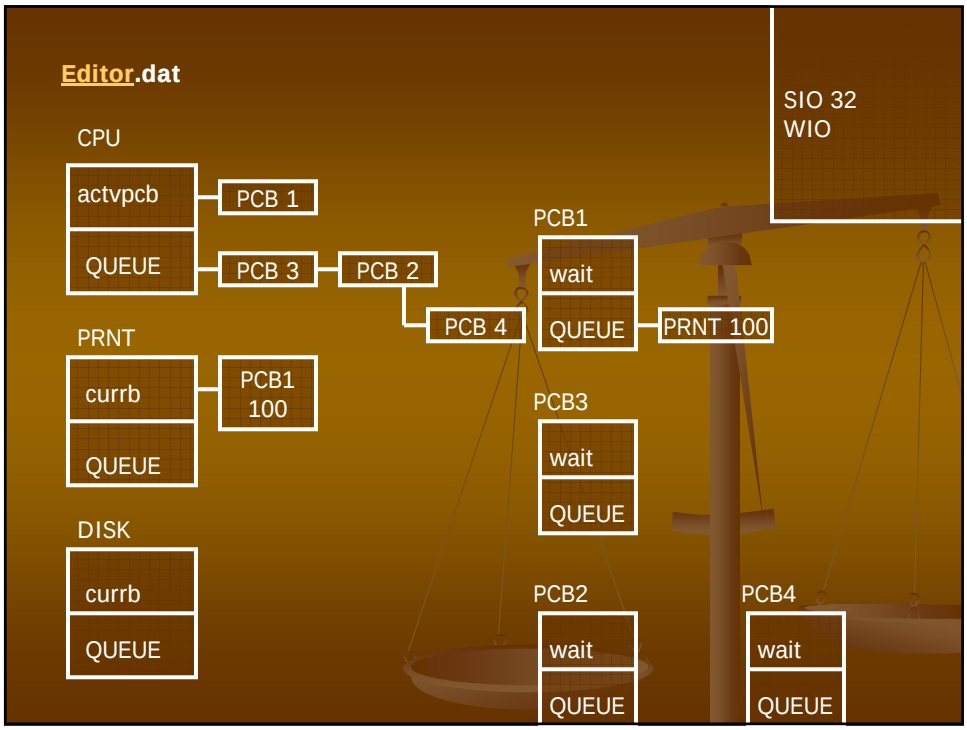

Editor.dat
CPU
actvpcb
PCB 1
QUEUE
PRNT
currb
QUEUE
DISK
currb
QUEUE
LOGON(U003)
LOGON(U002)
SIO 23
LOGON(U004)
PCB1
wait
QUEUE

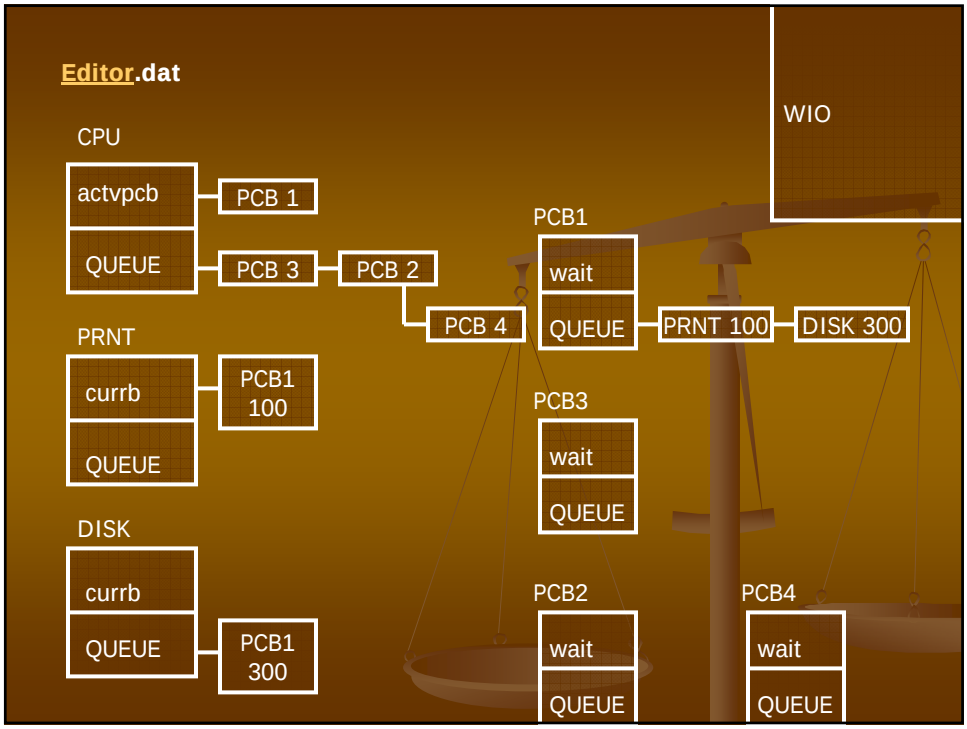

Editor.dat
CPU
actvpcb
PCB 1
QUEUE
PCB 3
PRNT
currb
QUEUE
DISK
currb
QUEUE
LOGON(U002)
SIO 23
LOGON(U004)
PCB1
wait
QUEUE
PCB3
wait
QUEUE

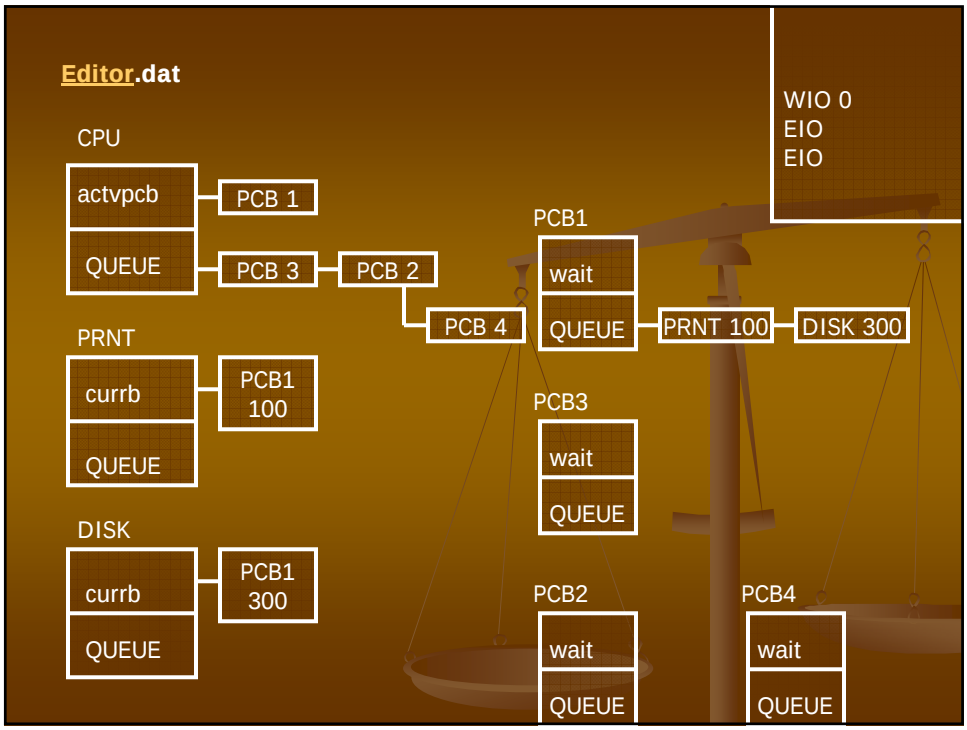

Editor.dat
CPU
actvpcb
PCB 1
QUEUE
PCB 3
PCB 2
PRNT
currb
QUEUE
DISK
currb
QUEUE
SIO 23
LOGON(U004)
PCB1
wait
QUEUE
PCB3
wait
QUEUE
PCB2
wait
QUEUE

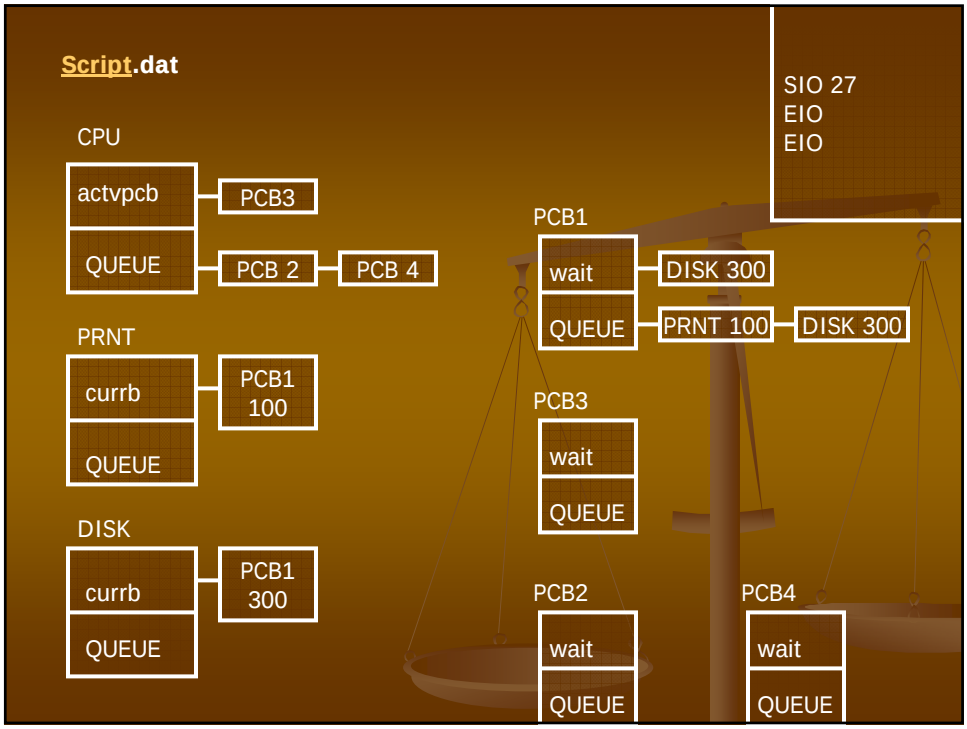

Editor.dat
CPU
actvpcb
PCB 1
QUEUE
PCB 3
PCB 2
PRNT
currb
QUEUE
PCB1
100
DISK
currb
QUEUE
LOGON(U004)
SIO 32
EIO
PCB1
wait
QUEUE
PRNT 100
PCB3
wait
QUEUE
PCB2
wait
QUEUE

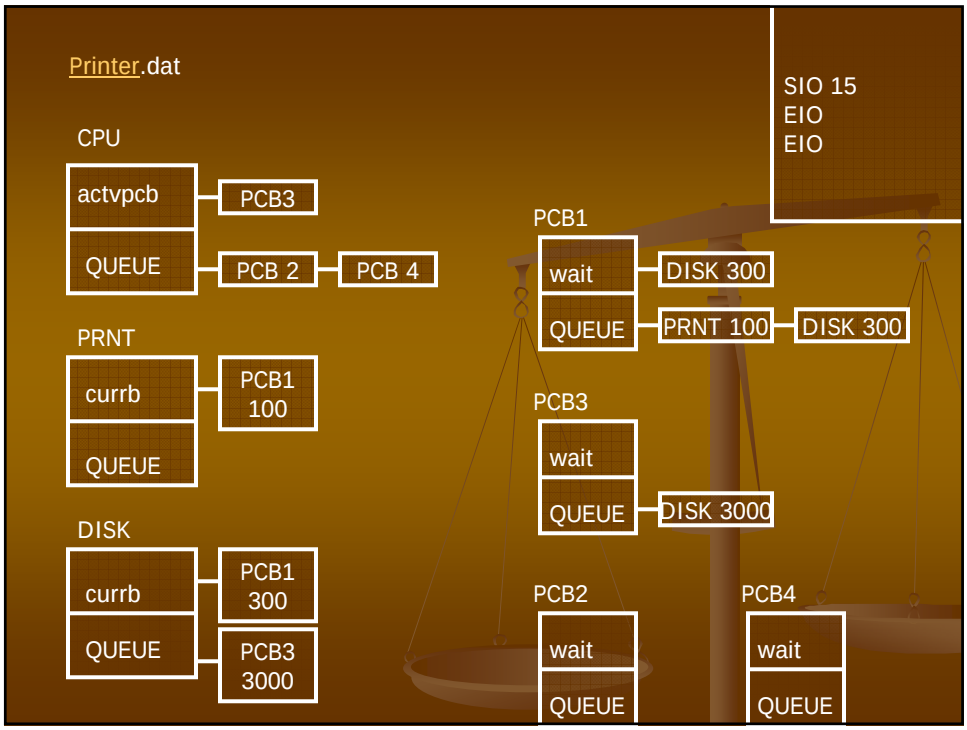

Editor.dat
LOGON(U004)
SIO 32
EIO
CPU
actvpcb
PCB 1
QUEUE
PCB 3
PCB 2
PRNT
currb
PCB1
100
QUEUE
DISK
currb
QUEUE
PCB1
wait
QUEUE
PRNT 100
PCB3
wait
QUEUE
PCB2
wait
QUEUE

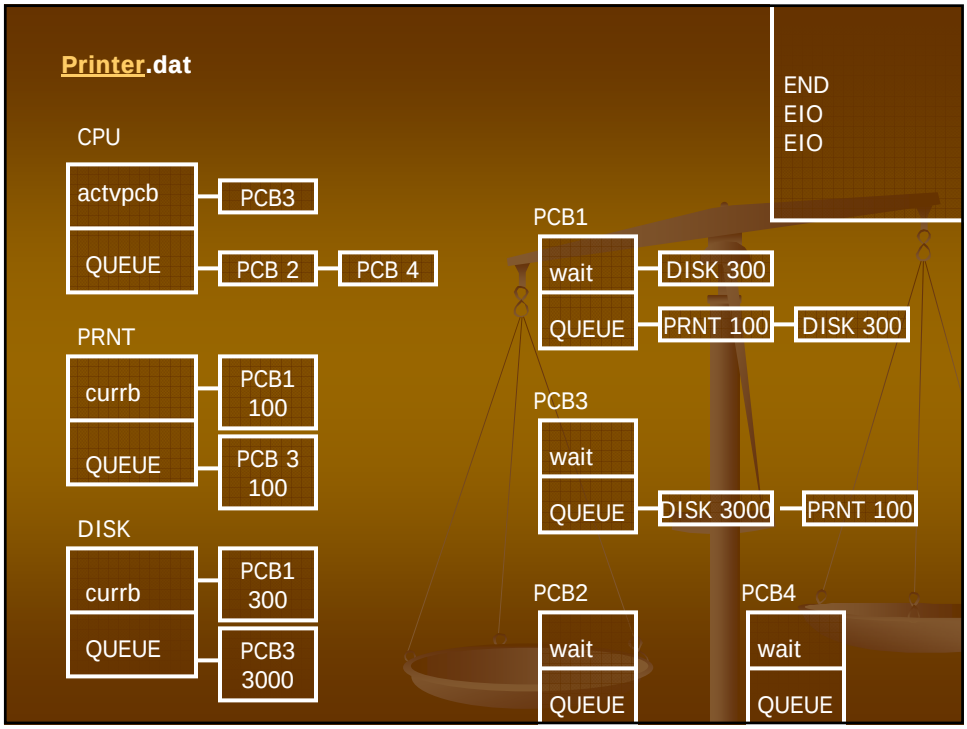

Editor.dat
SIO 32
WIO
CPU
actvpcb
PCB 1
QUEUE
PCB 3
PCB 2
PCB 4
PRNT
currb
PCB1
100
QUEUE
DISK
currb
QUEUE
PCB1
wait
QUEUE
PRNT 100
PCB3
wait
QUEUE
PCB2
wait
QUEUE
PCB4
wait
QUEUE

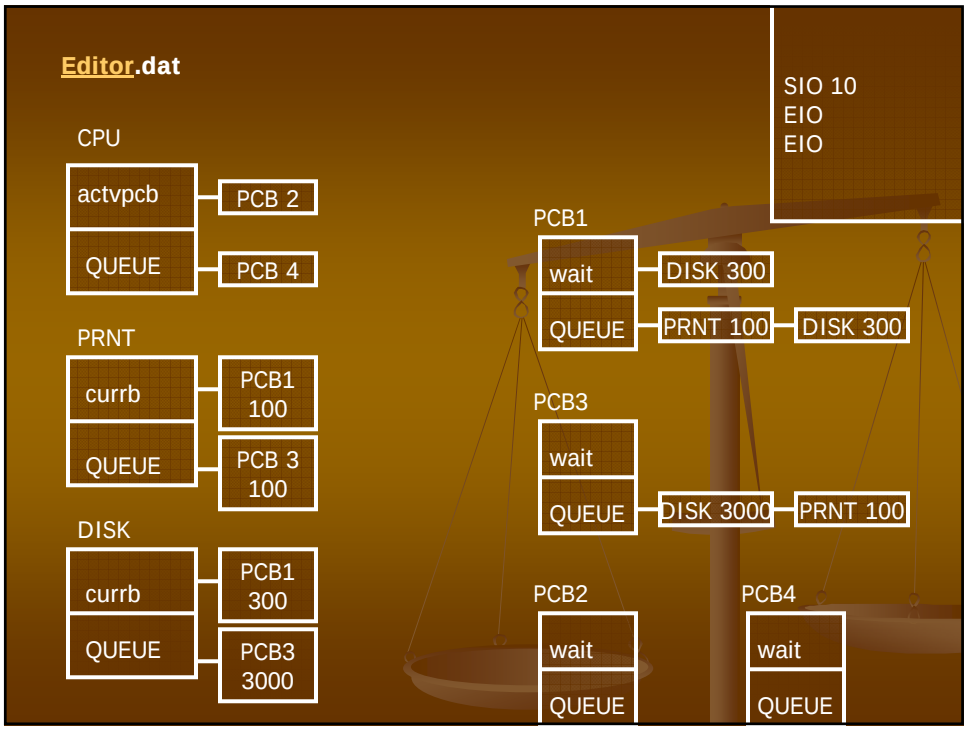

Editor.dat
WIO
CPU
actvpcb
PCB 1
QUEUE
PCB 3
PCB 2
PCB 4
PCB1
wait
QUEUE
PRNT 100
DISK 300
PRNT
currb
PCB1
100
QUEUE
PCB3
wait
QUEUE
DISK
currb
QUEUE
PCB1
300
PCB2
wait
QUEUE
PCB4
wait
QUEUE

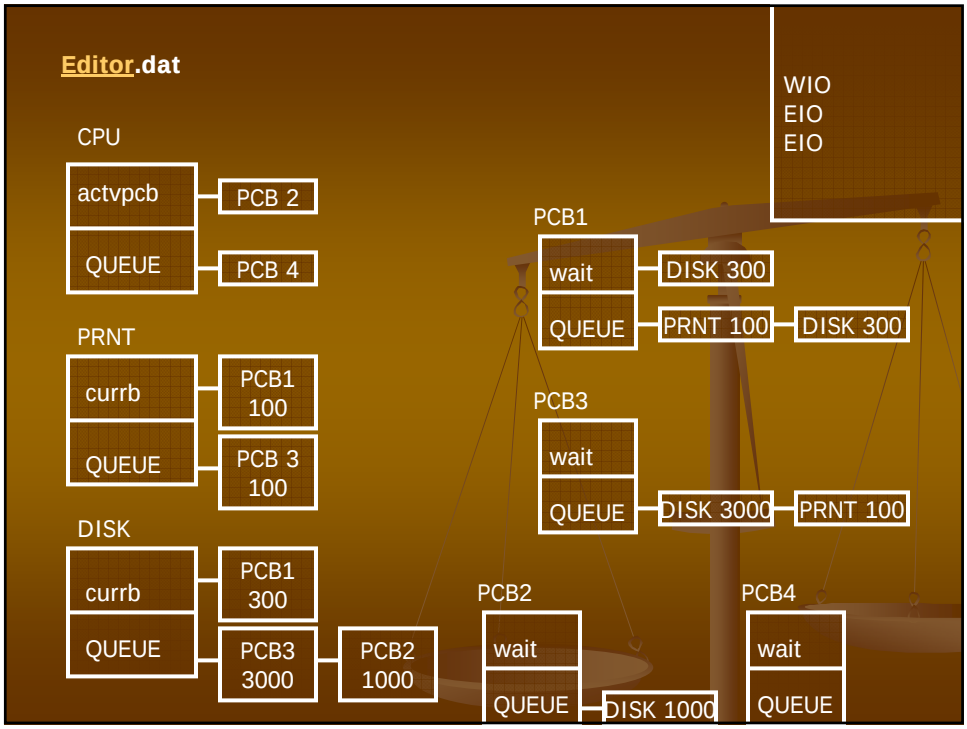

Editor.dat
WIO 0
EIO
EIO
CPU
actvpcb
PCB 1
QUEUE
PCB 3
PCB 2
PCB 4
PCB1
wait
QUEUE
PRNT 100
DISK 300
PRNT
currb
PCB1
100
QUEUE
PCB3
wait
QUEUE
DISK
currb
QUEUE
PCB1
300
PCB2
wait
QUEUE
PCB4
wait
QUEUE

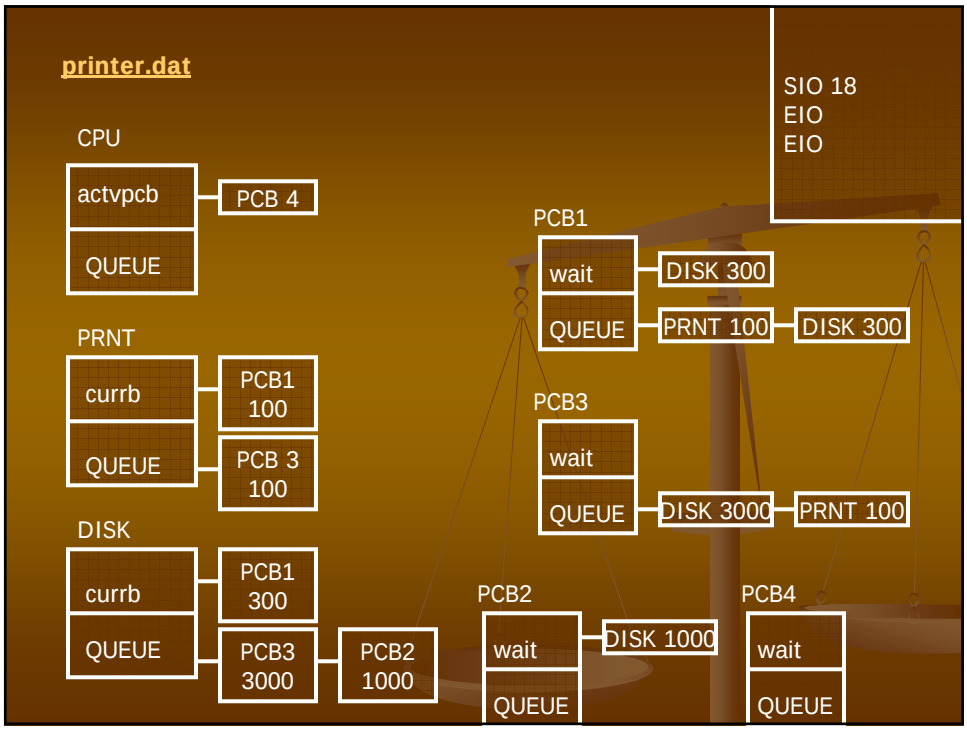

Script.dat
SIO 27
EIO
EIO
CPU
actvpcb
PCB3
QUEUE
PCB 2
PCB 4
PRNT
currb
PCB1
100
QUEUE
DISK
currb
PCB1
300
QUEUE
PCB1
wait
DISK 300
QUEUE
PRNT 100
DISK 300
PCB3
wait
QUEUE
PCB2
wait
QUEUE
PCB4
wait
QUEUE

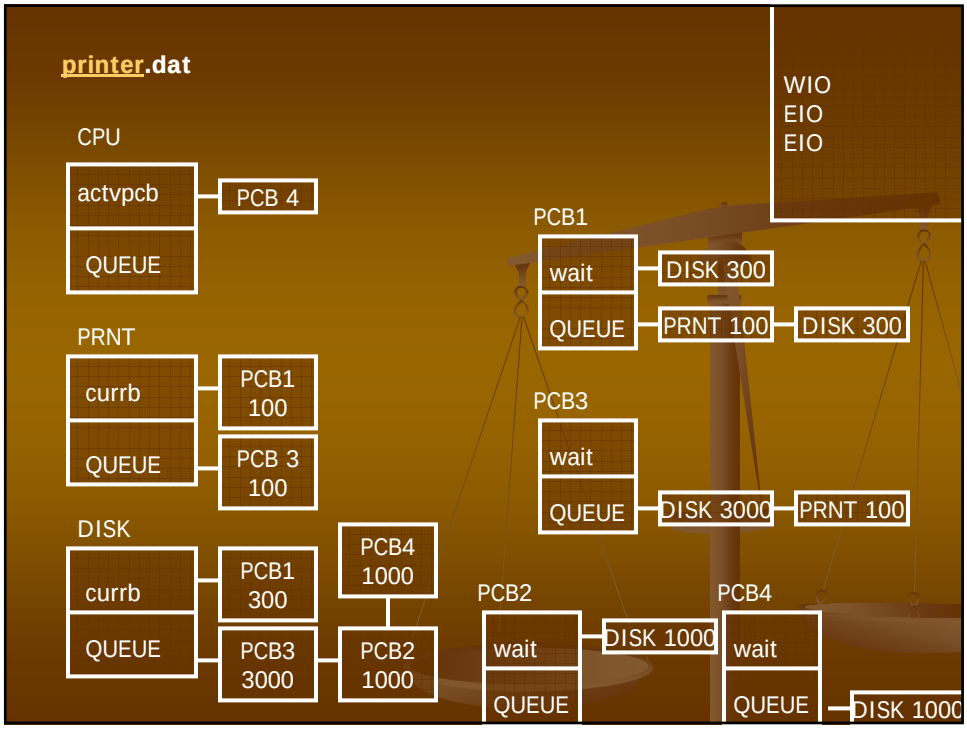

Printer.dat
SIO 15
EIO
EIO
CPU
actvpcb
PCB3
QUEUE
PCB 2
PCB 4
PRNT
currb
PCB1
100
QUEUE
DISK
currb
PCB1
300
QUEUE
PCB3
3000
PCB1
wait
DISK 300
QUEUE
PRNT 100
DISK 300
PCB3
wait
QUEUE
DISK 3000
PCB2
wait
QUEUE
PCB4
wait
QUEUE

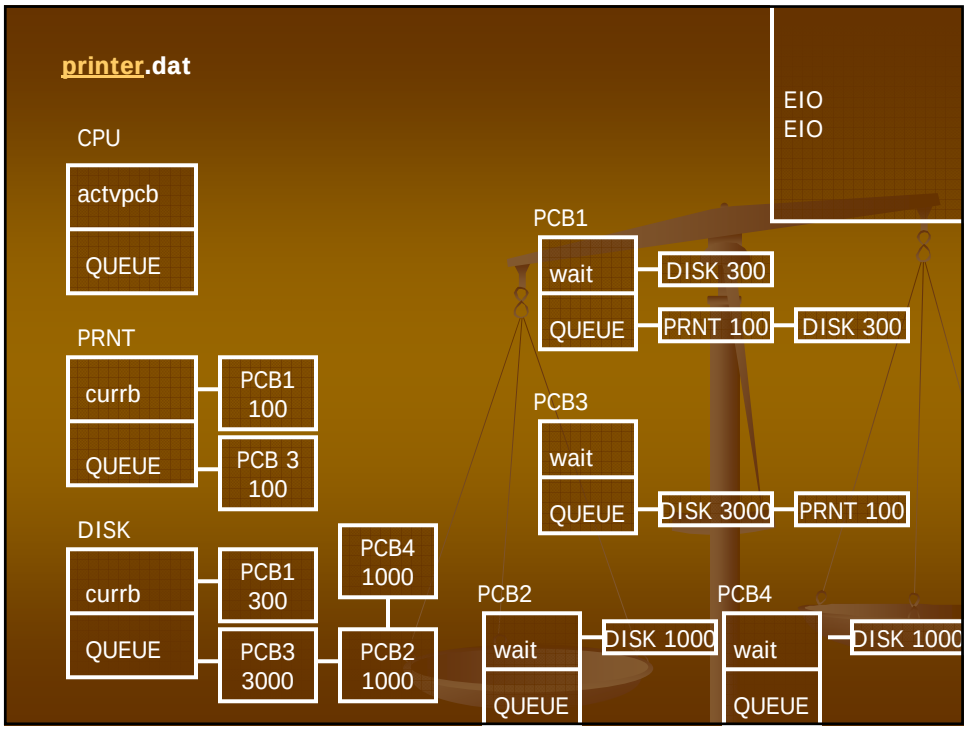

Printer.dat
END
EIO
EIO
CPU
actvpcb
PCB3
QUEUE
PCB 2
PCB 4
PRNT
currb
PCB1
100
QUEUE
PCB 3
100
DISK
currb
PCB1
300
QUEUE
PCB3
3000
PCB1
wait
DISK 300
QUEUE
PRNT 100
DISK 300
PCB3
wait
QUEUE
DISK 3000
PRNT 100
PCB2
wait
QUEUE
PCB4
wait
QUEUE

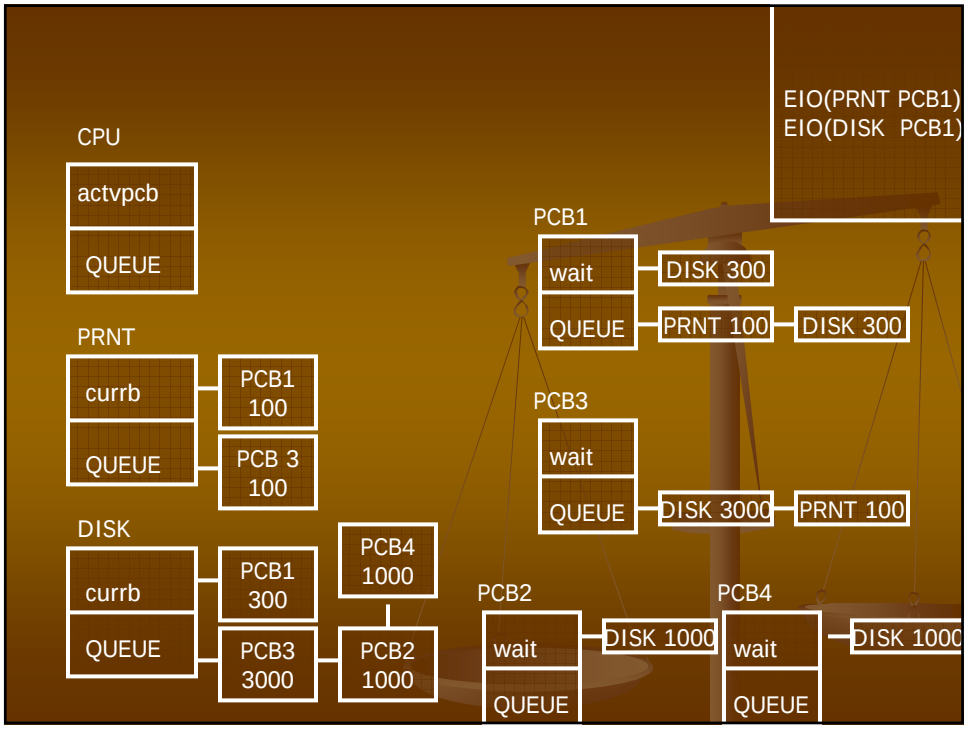

Editor.dat
SIO 10
EIO
EIO
CPU
actvpcb
PCB 2
QUEUE
PCB 4
PRNT
currb
PCB1
100
QUEUE
PCB 3
100
DISK
currb
PCB1
300
QUEUE
PCB3
3000
PCB1
wait
DISK 300
QUEUE
PRNT 100
DISK 300
PCB3
wait
QUEUE
DISK 3000
PRNT 100
PCB2
wait
QUEUE
PCB4
wait
QUEUE

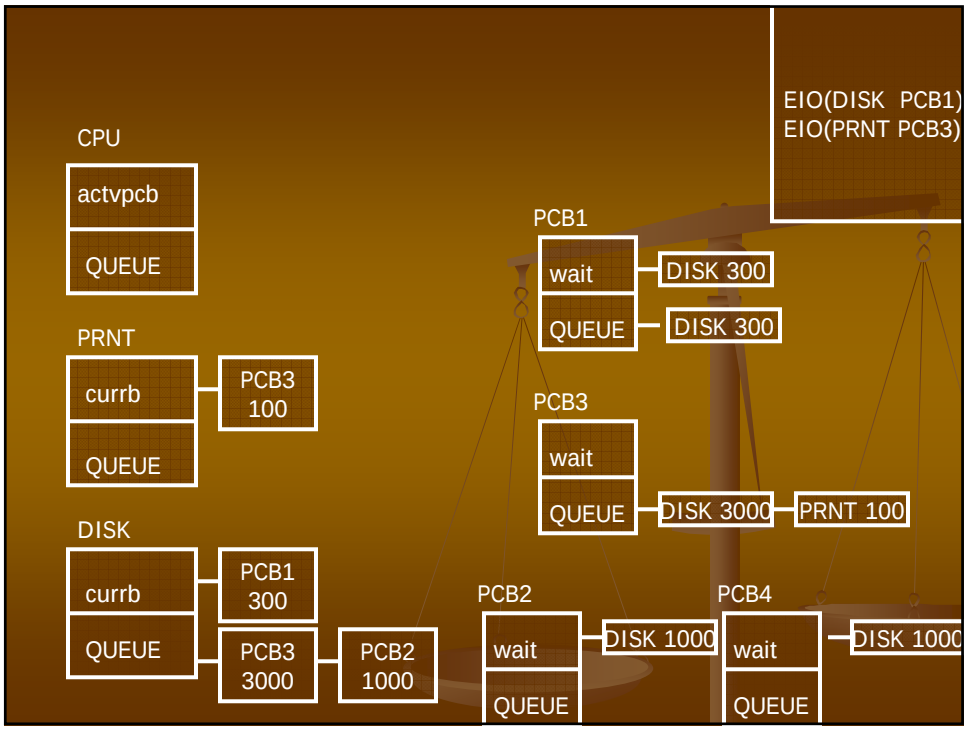

Editor.dat
WIO
EIO
EIO
CPU
actvpcb
PCB 2
QUEUE
PCB 4
PRNT
currb
PCB1
100
QUEUE
PCB 3
100
DISK
currb
PCB1
300
QUEUE
PCB3
3000
PCB2
1000
PCB1
wait
DISK 300
QUEUE
PRNT 100
DISK 300
PCB3
wait
QUEUE
DISK 3000
PRNT 100
PCB2
wait
QUEUE
DISK 1000
PCB4
wait
QUEUE

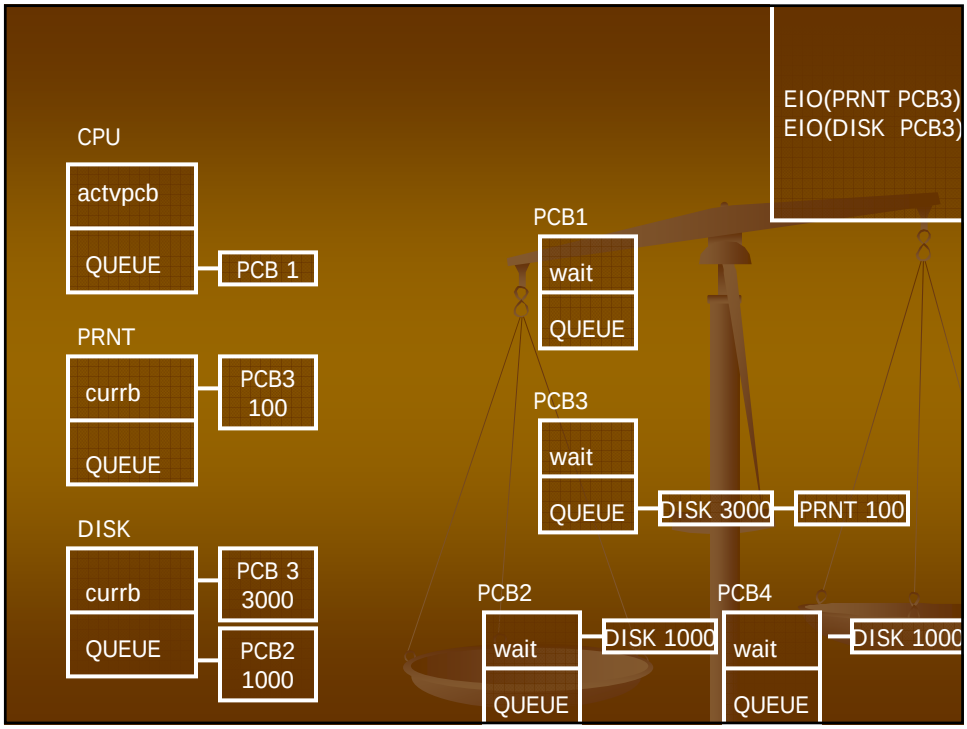

printer.dat
SIO 18
EIO
EIO
CPU
actvpcb
PCB 4
QUEUE
PRNT
currb
PCB1
100
QUEUE
PCB 3
100
DISK
currb
PCB1
300
QUEUE
PCB3
3000
PCB2
1000
PCB1
wait
DISK 300
QUEUE
PRNT 100
DISK 300
PCB3
wait
QUEUE
DISK 3000
PRNT 100
PCB2
wait
DISK 1000
QUEUE
PCB4
wait
QUEUE

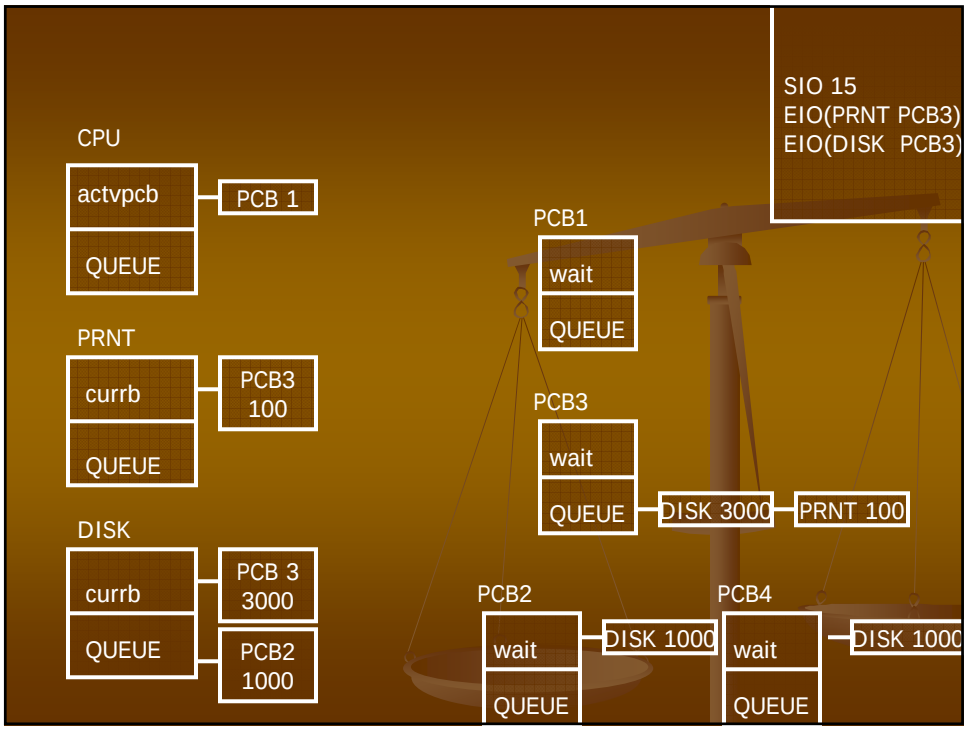

printer.dat
WIO
EIO
EIO
CPU
actvpcb
PCB 4
QUEUE
PRNT
currb
PCB1
100
QUEUE
PCB 3
100
DISK
currb
PCB1
300
QUEUE
PCB3
3000
PCB4
1000
PCB2
1000
PCB1
wait
DISK 300
QUEUE
PRNT 100
DISK 300
PCB3
wait
QUEUE
DISK 3000
PRNT 100
PCB2
wait
DISK 1000
QUEUE
PCB4
wait
QUEUE
DISK 1000

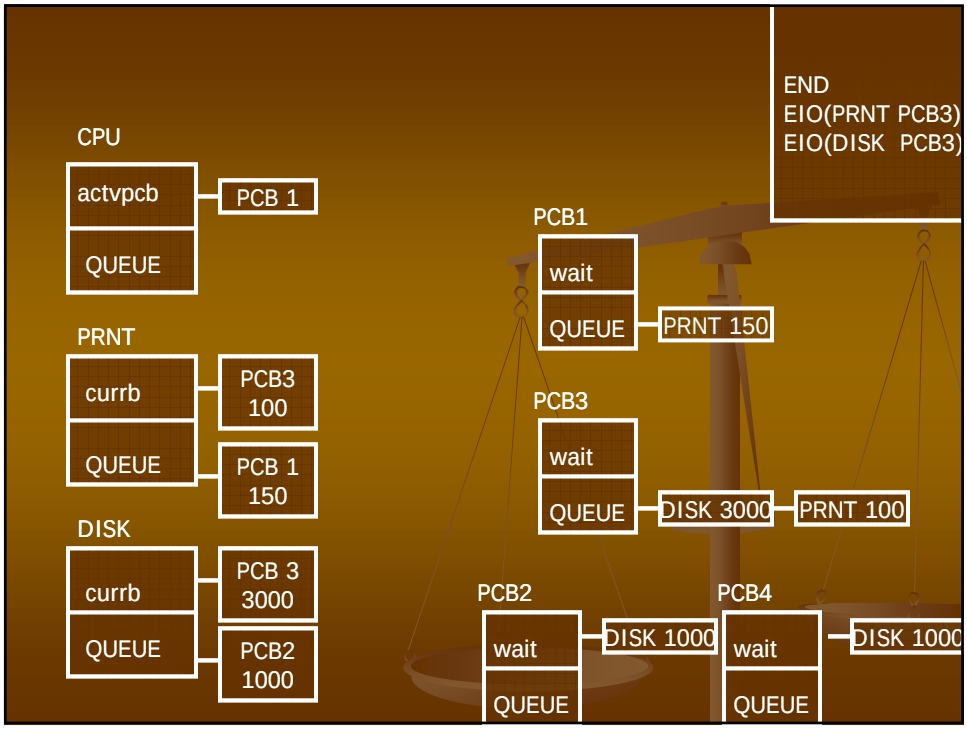

printer.dat
EIO
EIO
CPU
actvpcb
QUEUE
PRNT
currb
PCB1
100
QUEUE
PCB 3
100
DISK
currb
PCB1
300
QUEUE
PCB3
3000
PCB4
1000
PCB2
1000
PCB1
wait
DISK 300
QUEUE
PRNT 100
DISK 300
PCB3
wait
QUEUE
DISK 3000
PRNT 100
PCB2
wait
DISK 1000
QUEUE
PCB4
wait
QUEUE
DISK 1000

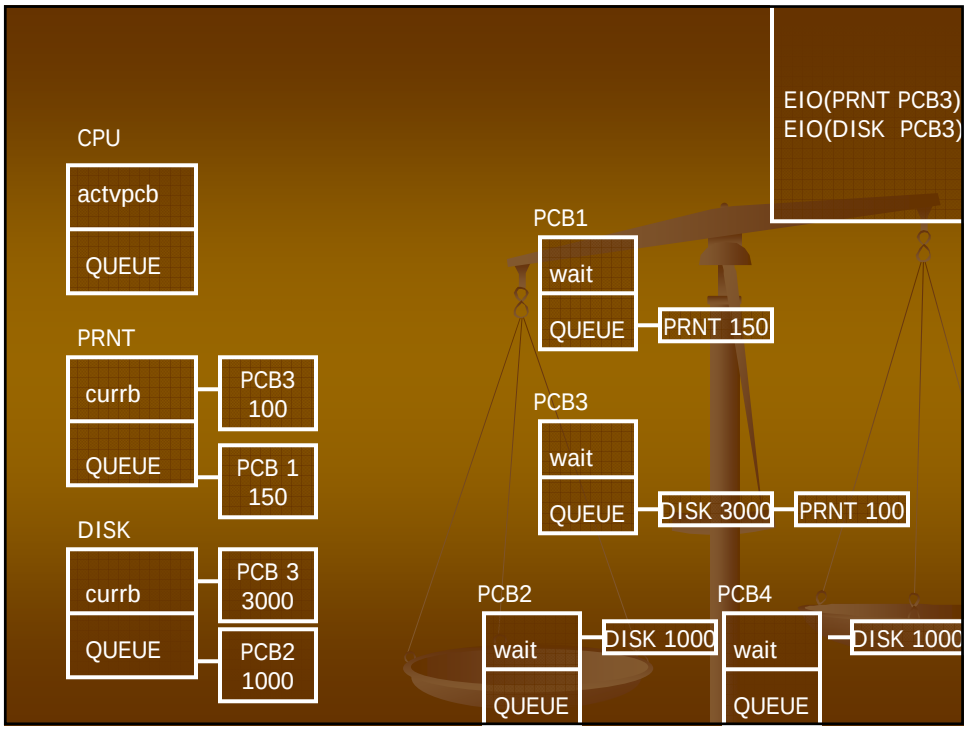

EIO(PRNT PCB1)
EIO(DISK PCB1)
CPU
actvpcb
QUEUE
PRNT
currb
QUEUE
PCB1
100
PCB 3
100
DISK
currb
QUEUE
PCB1
300
PCB3
3000
PCB4
1000
PCB2
1000
PCB1
wait
QUEUE
DISK 300
PRNT 100
DISK 300
PCB3
wait
QUEUE
DISK 3000
PRNT 100
PCB2
wait
QUEUE
DISK 1000
PCB4
wait
QUEUE
DISK 1000

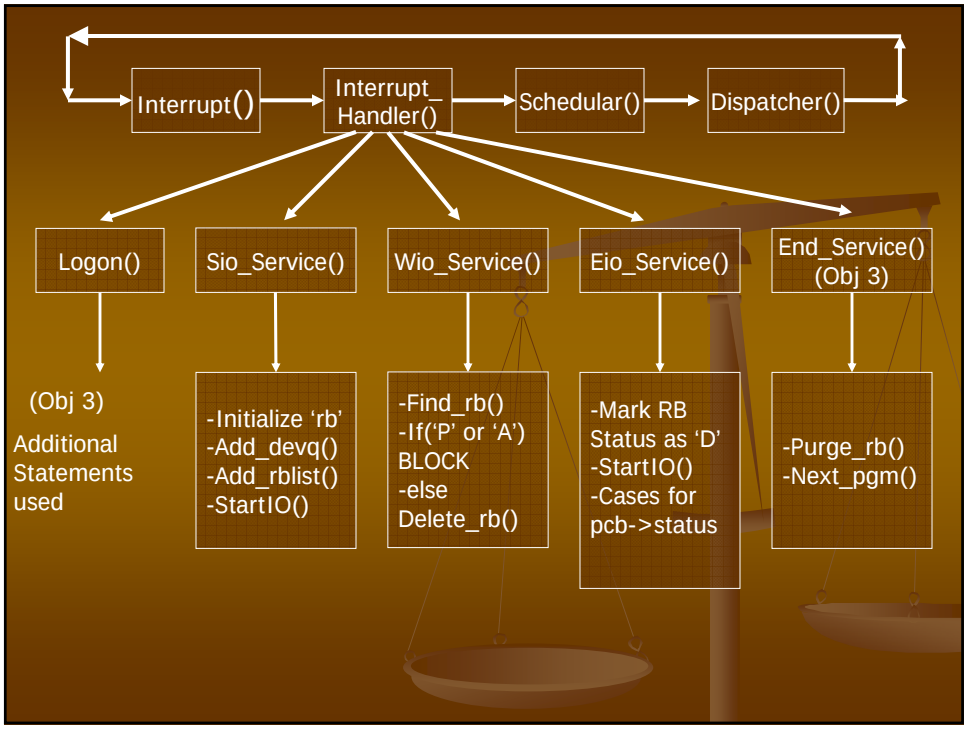

EIO(DISK PCB1)
EIO(PRNT PCB3)
CPU
actvpcb
QUEUE
PRNT
currb
QUEUE
PCB3
100
DISK
currb
QUEUE
PCB1
300
PCB3
3000
PCB2
1000
PCB1
wait
QUEUE
DISK 300
DISK 300
PCB3
wait
QUEUE
DISK 3000
PRNT 100
PCB2
wait
QUEUE
DISK 1000
PCB4
wait
QUEUE
DISK 1000

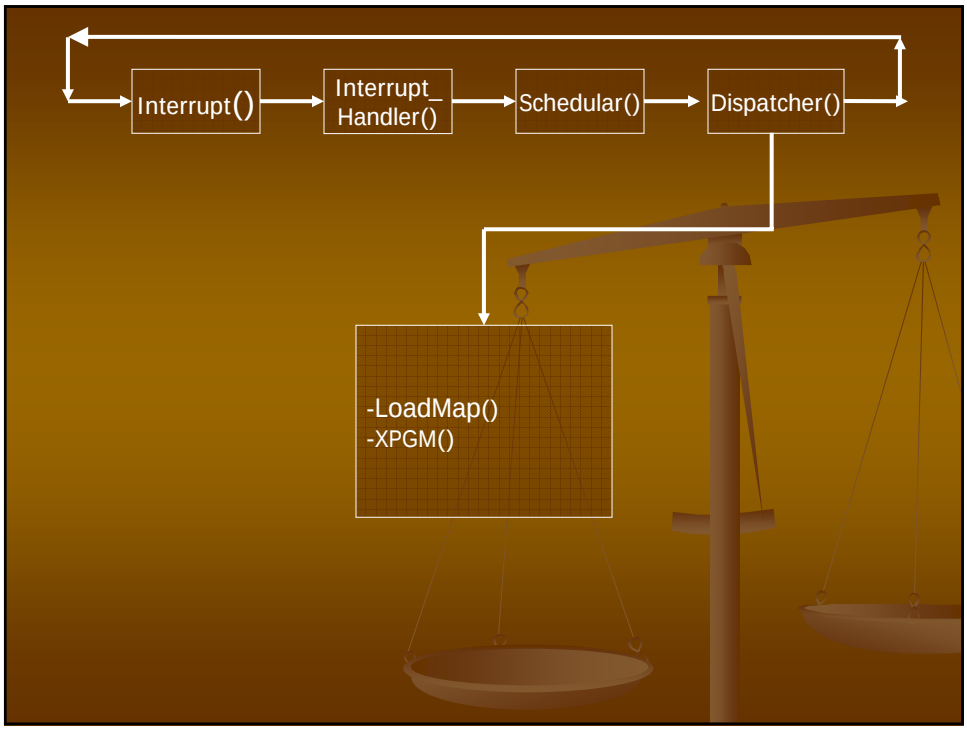

EIO(PRNT PCB3)
EIO(DISK PCB3)
CPU
actvpcb
QUEUE
PCB 1
PRNT
currb
QUEUE
PCB3
100
DISK
currb
QUEUE
PCB 3
3000
PCB2
1000
PCB1
wait
QUEUE
PCB3
wait
QUEUE
DISK 3000
PRNT 100
PCB2
wait
QUEUE
DISK 1000
PCB4
wait
QUEUE
DISK 1000


SIO 15
EIO(PRNT PCB3)
EIO(DISK PCB3)
CPU
actvpcb
QUEUE
PCB 1
PRNT
currb
QUEUE
PCB3
100
DISK
currb
QUEUE
PCB 3
3000
PCB2
1000
PCB1
wait
QUEUE
PCB3
wait
QUEUE
DISK 3000
PRNT 100
PCB2
wait
QUEUE
DISK 1000
PCB4
wait
QUEUE
DISK 1000


END
EIO(PRNT PCB3)
EIO(DISK PCB3)
CPU
actvpcb
PCB 1
QUEUE
PRNT
currb
PCB3
100
QUEUE
PCB 1
150
DISK
currb
PCB 3
3000
QUEUE
PCB2
1000
PCB1
wait
QUEUE
PRNT 150
PCB3
wait
QUEUE
DISK 3000
PRNT 100
PCB2
wait
DISK 1000
QUEUE
PCB4
wait
DISK 1000
QUEUE


EIO(PRNT PCB3)
EIO(DISK PCB3)
CPU
actvpcb
QUEUE
PRNT
currb
PCB3
100
QUEUE
PCB 1
150
DISK
currb
PCB 3
3000
QUEUE
PCB2
1000
PCB1
wait
QUEUE
PRNT 150
PCB3
wait
QUEUE
DISK 3000
PRNT 100
PCB2
wait
DISK 1000
QUEUE
PCB4
wait
DISK 1000
QUEUE


Interrupt()
Interrupt_ Handler()
Schedular()
Dispatcher()
Logon()
Sio_Service()
Wio_Service()
Eio_Service()
End_Service() (Obj 3)
(Obj 3)
Additional Statements used
-Initialize 'rb'
-Add_devq()
-Add_rblist()
-StartIO()
-Find_rb()
-If('P' or 'A') BLOCK
-else Delete_rb()
-Mark RB Status as 'D'
-StartIO()
-Cases for pcb->status
-Purge_rb()
-Next_pgm()


Interrupt()
Interrupt_ Handler()
Schedular()
Dispatcher()
-LoadMap()
-XPGM()

- Add_cpuq(struct pcb_type *pcb)
  - Create a pcb_list* node
  - and set pcb to the pcb_type pointer in the node
  - If the wait queue in CPU is empty, add the node to head of the wait queue. Update the tail in CPU
  - Otherwise, add the node to the end of the wait queue in the CPU. Update the tail.

- Add_devq(int dev, struct rb_type *rb)
  - Create a rb_list* node
  - and set rb to the rb_type pointer in the node
  - Record the time when the rb is queued
  - If the device is empty, add the node to head of the wait queue in device. Update the tail in device
  - Otherwise, add the node to the end of the wait queue in the device. Update the tail.

- Add_rblist(struct pcb_type *pcb, struct rb_type *rb)
  - Create a rb_list* node
  - and set rb to the rb_type pointer in the node
  - Record the time when the rb is queued
  - If the rb_list queue is empty in pcb, add the node to head of the rb_list queue in pcb. Update the tail in pcb
  - Otherwise, add the node to the end of the rb_list queue in the pcb. Update the tail.

- Struct pcb_type* scheduler()
  - If the ready queue in CPU is empty, return
  - Update the number of processes served by the CPU
  - FCFS:
    - Grab the first node in the list
    - Record when the process became ACTIVE
    - Calculate time process was READY and increment CPU qwait time

- Dispatcher()
  - Prepares the scheduled program for execution and then transfers control to it
  - LoadMap (CPU.avtvpcb->segtable, CPU.actvpcb->segtab_len)
  - XPGM(&CPU.actvpcb->cpu_save);

- Sio_service() (Non Blocking function)
  - Services requests to Start IO
  - Allocates an I/O request block (rb)
  - Sets rb->status to “Pending”
  - Sets rb->pcb to Termtable[AGENT –1]
  - Sets rb->dev to devtable index of requested device
  - dev_addr = (int)MEMMAP[rb->pcb-> cpu_save. pc.segment] .membase + rb->pcb- >cpu_save.pc.offset – 1;
    rb->dev = (int) MEM[dev_addr]. opcode) – OPCDSIZE – 1;
    EG: SIO 35,PRNT 200, WIO 24

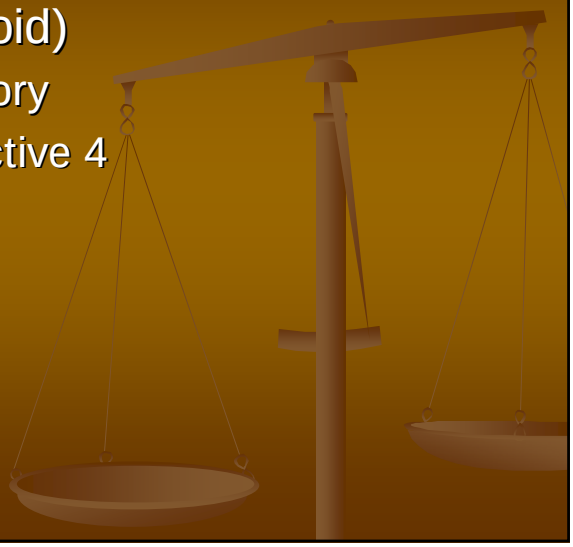
- Sio_service() (Non Blocking function)
  - Set rb->queuet to the value of CLOCK
  - Set rb->bytes
  - Set rb-> reqid to logical address of Device instruction

    rb-> reqid.segment = rb->pcb->cpu_save.pc.segment ;

    rb-> reqid.offset = rb->pcb->cpu_save.pc.offset – 1;
  - Add_devq();
  - Add_rblist();
  - Attempt to initiate an operation on the requested device – StartIO();
  - Set CPU_SW = 1 and SCHED_SW = 0;

- Wio_Service(void) (Blocking Function)
  - Locate REQ instruction to retrieve address field.
  - Call Find_rb() to to get status of IO operation waited upon.
  - If status is ‘P’ or ‘A’, BLOCK pcb.
  - Set CPU_SW = SCHED_SW = 1
  - Calculate ACTIVE time for process and Busy time for CPU
  - Record the time the process what blocked and return.
  - Else process can continue running.
    - Call Delete_rb() to deallocate the IORB.
    - Set CPU_SW = 1 , SCHED_SW = 0;

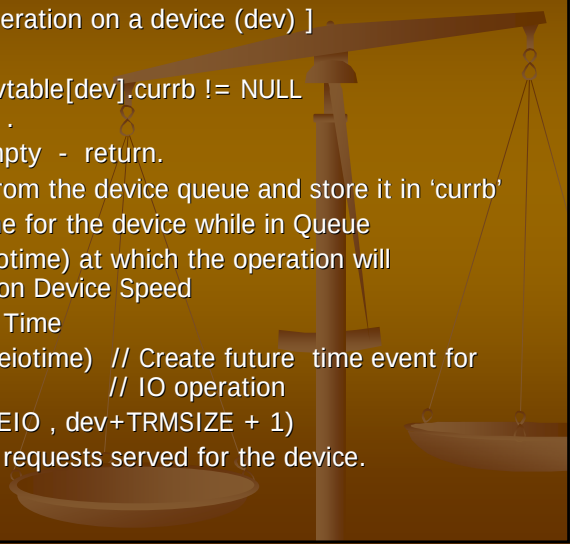
- Eio_Service(void)
  - Self explanatory
  - See the objective 4



Start_IO(int dev)

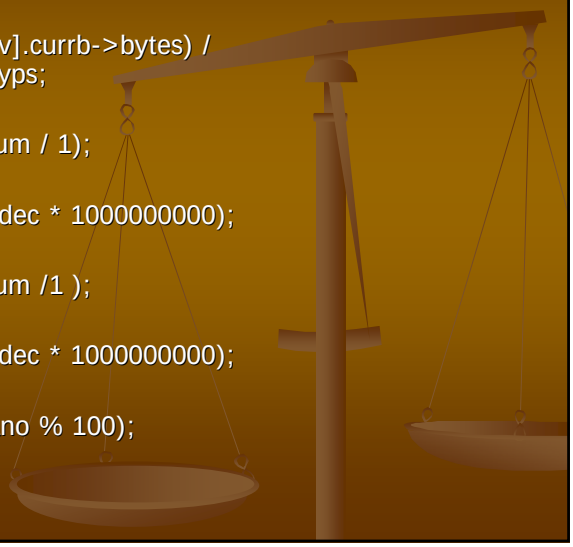
[Attempts to start an operation on a device (dev)]

- If Device is busy : devtable[dev].currb != NULL
  - return .
- If Device Queue is empty - return.
- Remove the next rb from the device queue and store it in 'currb'
- Update Total Wait time for the device while in Queue
- Calculate the time (eiotime) at which the operation will terminate depending on Device Speed
- Update Device's Busy Time
- Add_time(&CLOCK, &eiotime) // Create future time event for // IO operation
  Add_event(&eiotime, EIO , dev+TRMSIZE + 1)
- Update number of IO requests served for the device.



```
double num;
Num =((double)devtable[dev].currb->bytes) /
    ((double)devtable[dev].byps;
If(num >= 1)
    sec = (unsigned long) (num / 1);
    dec = num – sec;
    nano = (unsigned long) (dec * 1000000000);
else
    sec = (unsigned long) (num /1 );
    dec = num;
    nano = (unsigned long) (dec * 1000000000);

NOTE :  nano =  nano – (nano % 100);
```



- **Find_rb()**

  [Searches RB list of PCB to locate an RB]

- **Delete_rb()**

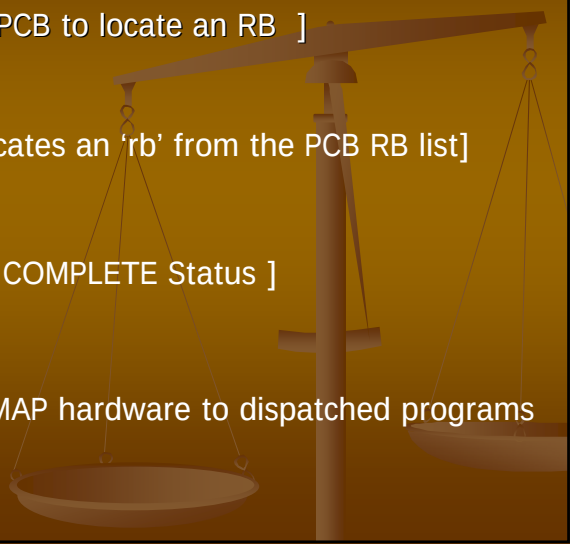
  [Deletes and deallocates an ‘rb’ from the PCB RB list]

- **Purge_rb()**

  [Purges all RB’s with COMPLETE Status]

- **LoadMap()**

  [Initialises the MEMMAP hardware to dispatched programs segment table]



Objective 5

- Calc_stats(void)

[The Function computes all the simulation statistics and stores them in the appropriate variables and data structures for display. This function is called by Wrapup()]

for i = 0 to TRMSIZE // I.e for each PCB {

- Total Processing time - TOTLOGON
- Total Job blocked time - TOTBLKED
- Total Job Wait time - TOTWAIT
- Total Job Execution Time – TOTRUN
- Efficiency for each process –

temp = total blocked time + total Run time

termtable[i]->effciency = 100.0 * Divd_time(&temp, &termtable[i]->tlogon);

}

for i= 0 to DEVSIZE

{

[Calculate Response time for all devices]

• temp = Busy time + Qwait time

•Ave_time(&devtable[i].response,&temp,&devtable[i].served);

[Calculate idle time]

• devtable[i].idle.seconds = CLOCK.seconds

• devtable[i].idle.nanosec = CLOCK.nanosec

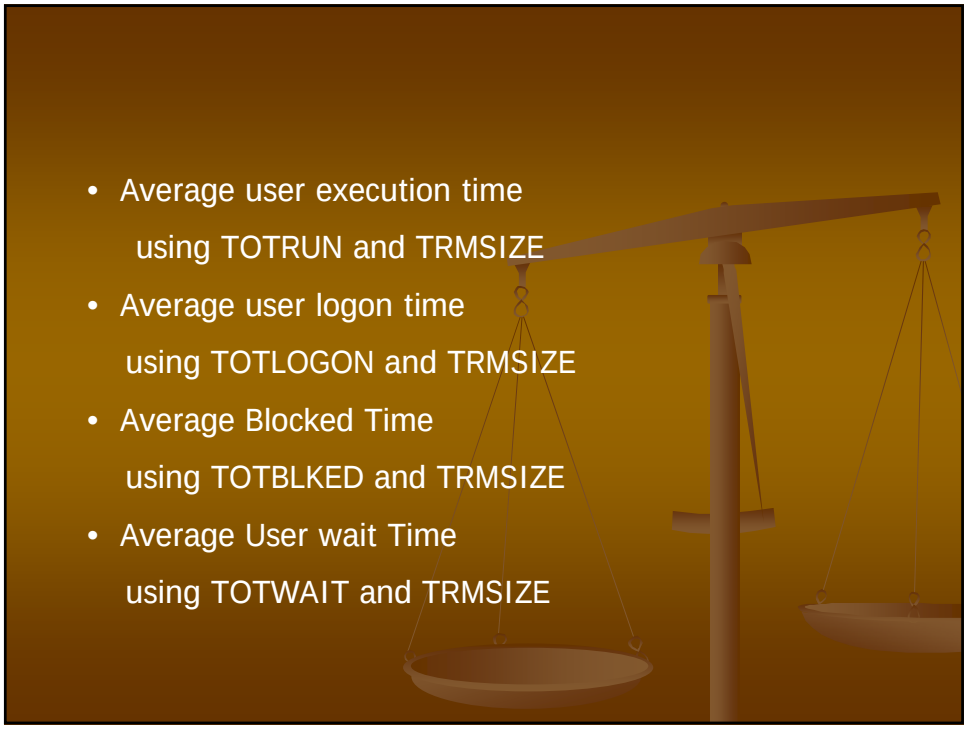
Diff_time(&devtable[i].busy, &devtable[i].idle)

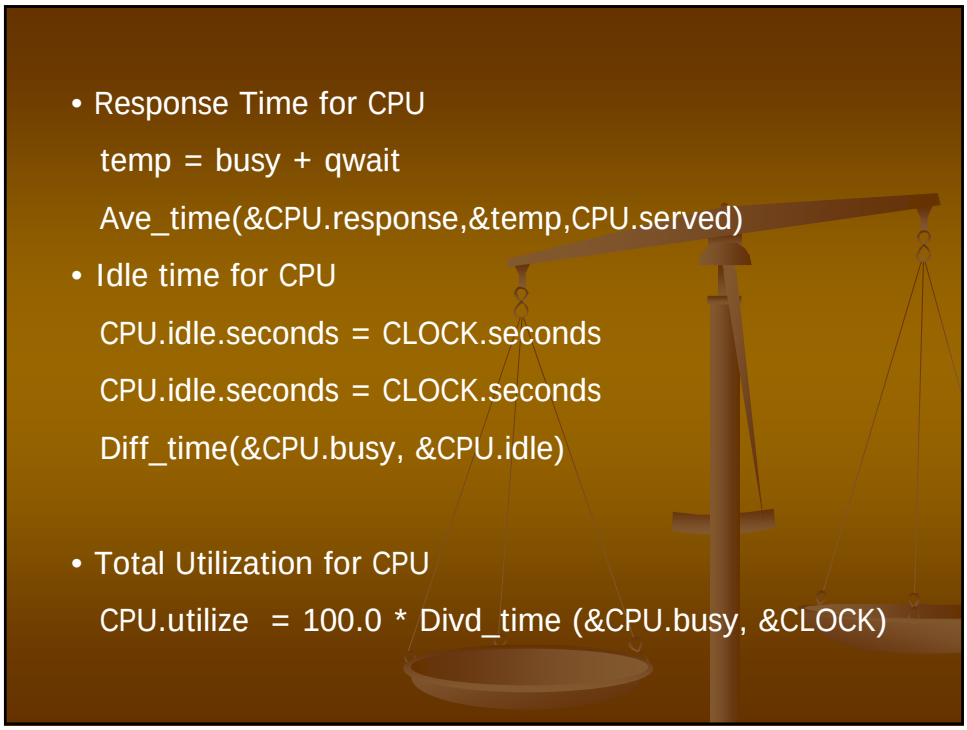
[Utilization]

Devtable[i].utilize =

100.0 * Divd_time(&devtable[i].busy,&CLOCK)

}


• Average user execution time
using TOTRUN and TRMSIZE
• Average user logon time
using TOTLOGON and TRMSIZE
• Average Blocked Time
using TOTBLKED and TRMSIZE
• Average User wait Time
using TOTWAIT and TRMSIZE


• Response Time for CPU
temp = busy + qwait
Ave_time(&CPU.response,&temp,CPU.served)
• Idle time for CPU
CPU.idle.seconds = CLOCK.seconds
CPU.idle.seconds = CLOCK.seconds
Diff_time(&CPU.busy, &CPU.idle)
• Total Utilization for CPU
CPU.utilize = 100.0 * Divd_time (&CPU.busy, &CLOCK)