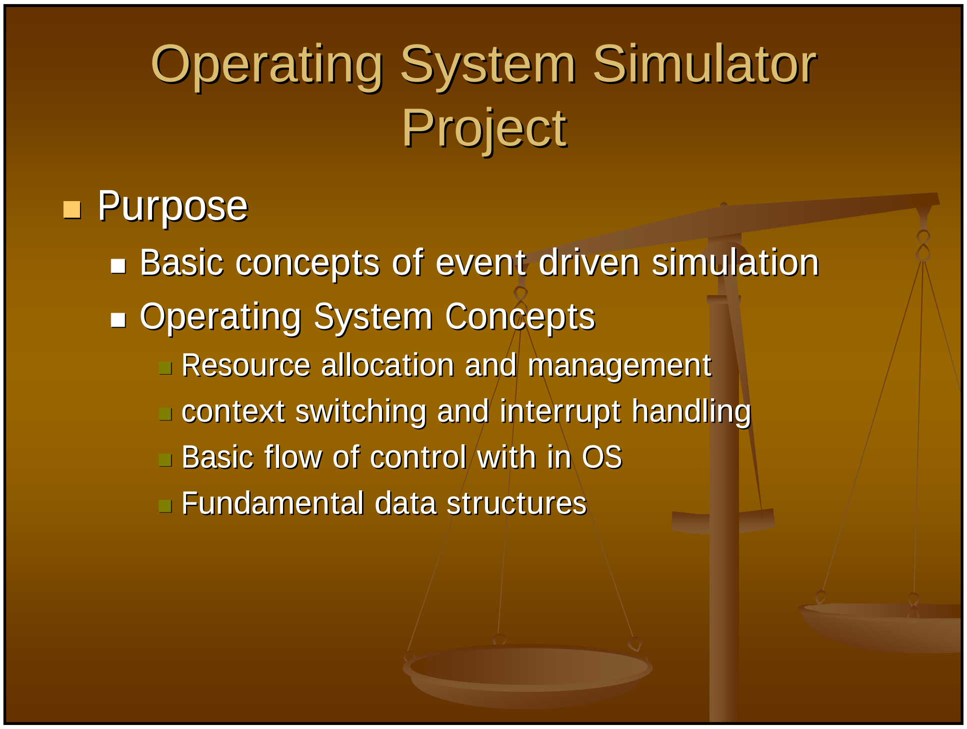
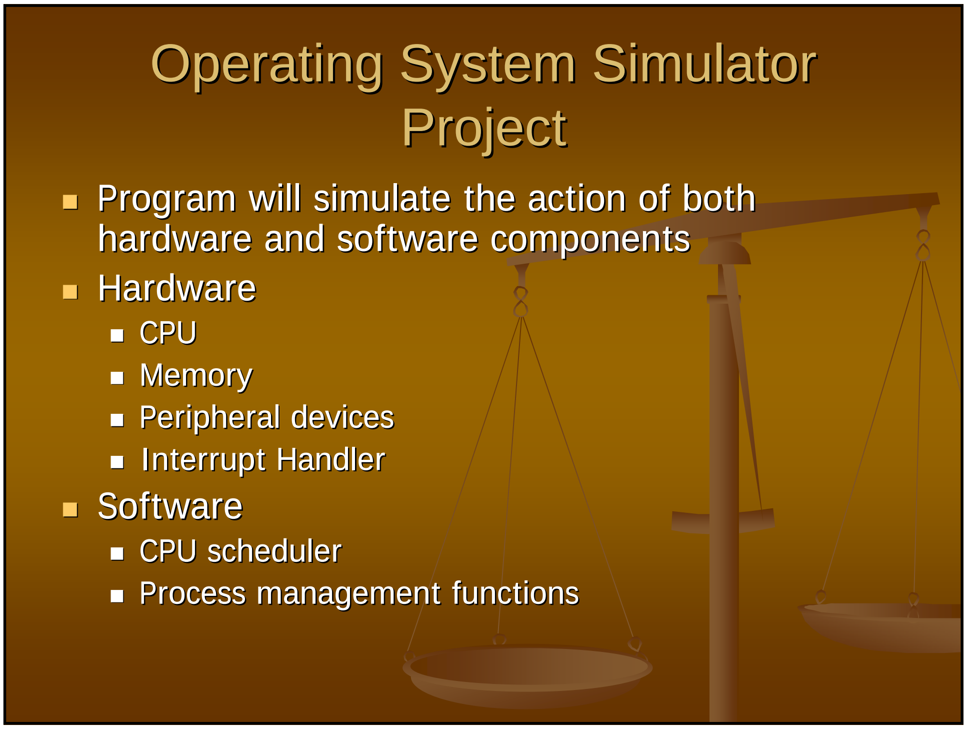


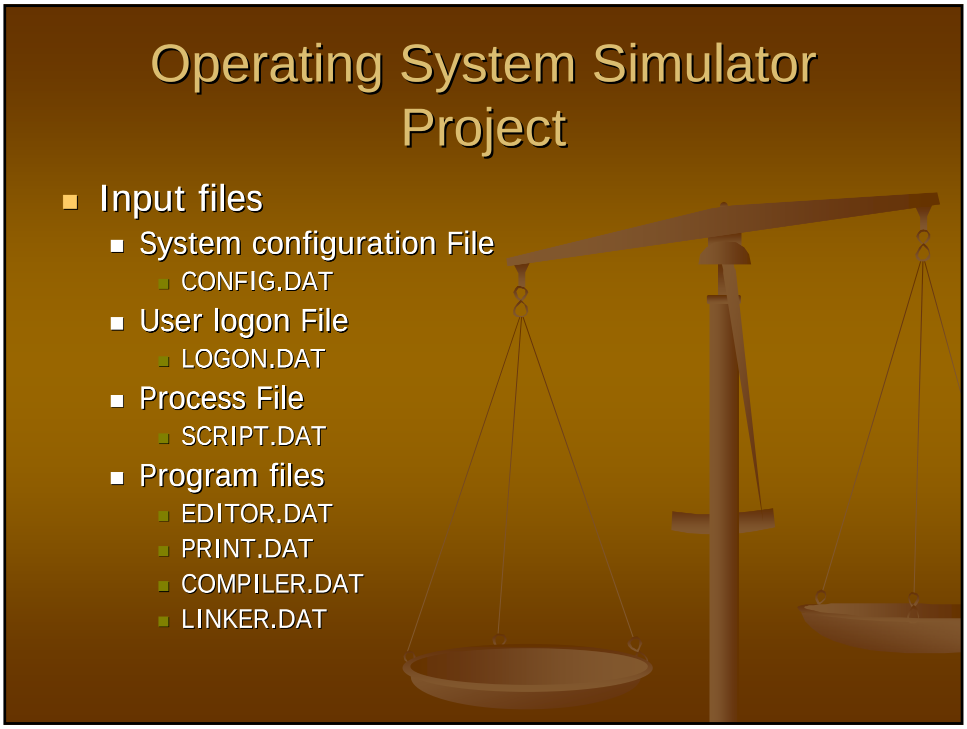
Operating System Simulator Project

- Purpose
  - Basic concepts of event driven simulation
  - Operating System Concepts
    - Resource allocation and management
    - context switching and interrupt handling
    - Basic flow of control with in OS
    - Fundamental data structures



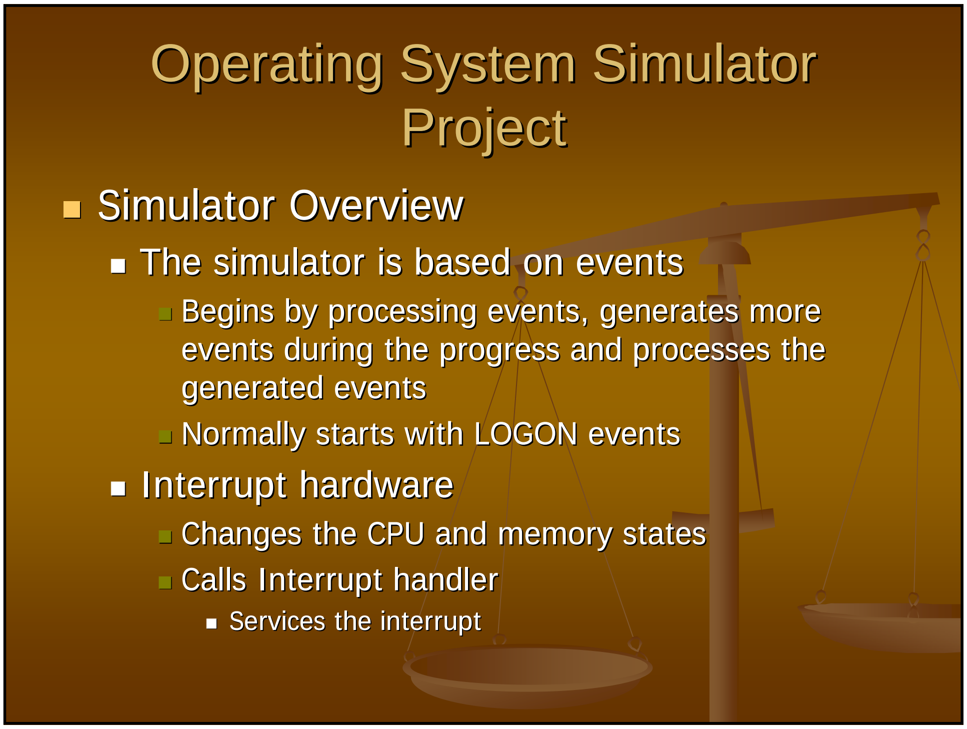
Operating System Simulator Project

- Program will simulate the action of both hardware and software components
- Hardware
  - CPU
  - Memory
  - Peripheral devices
  - Interrupt Handler
- Software
  - CPU scheduler
  - Process management functions



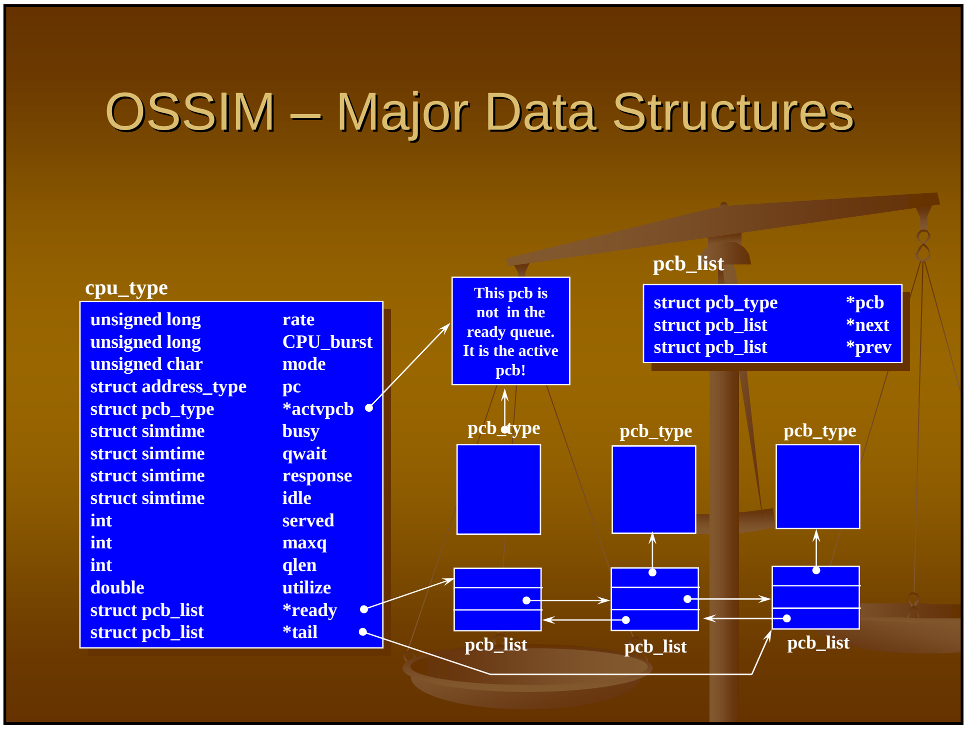
Operating System Simulator Project

- Input files
  - System configuration File
    - CONFIG.DAT
  - User logon File
    - LOGON.DAT
  - Process File
    - SCRIPT.DAT
  - Program files
    - EDITOR.DAT
    - PRINT.DAT
    - COMPILER.DAT
    - LINKER.DAT



Operating System Simulator Project

- Simulator Overview
  - The simulator is based on events
    - Begins by processing events, generates more events during the progress and processes the generated events
    - Normally starts with LOGON events
  - Interrupt hardware
    - Changes the CPU and memory states
    - Calls Interrupt handler
      - Services the interrupt



OSSIM – Major Data Structures

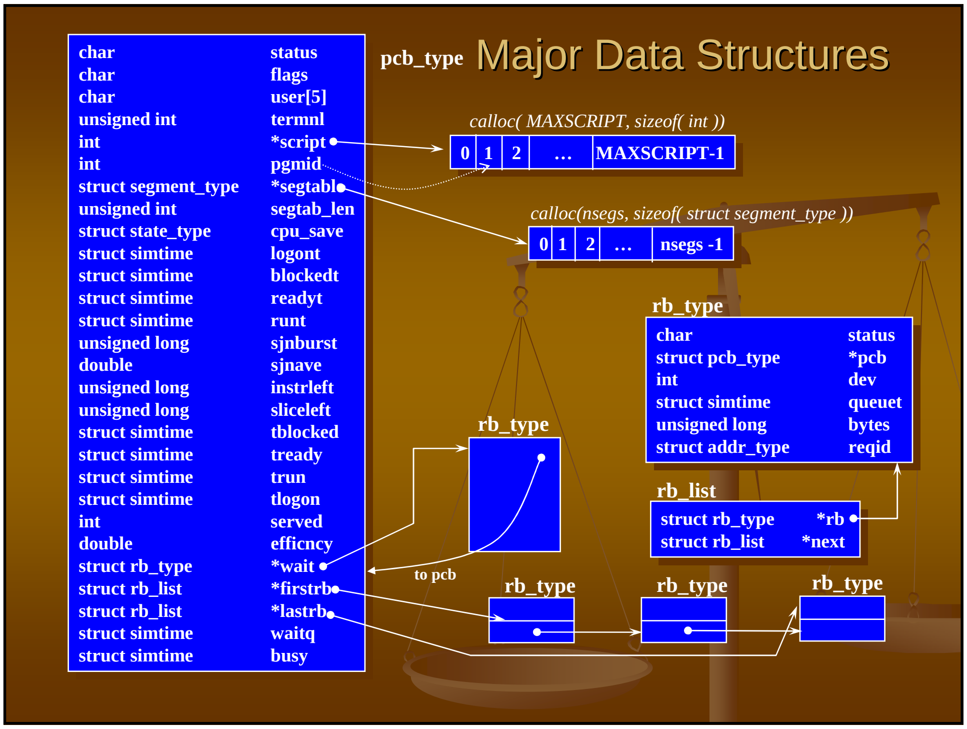
cpu_type

unsigned long	rate
unsigned long	CPU_burst
unsigned char	mode
struct address_type	pc
struct pcb_type	*actvpcb
struct simtime	busy
struct simtime	qwait
struct simtime	response
struct simtime	idle
int	served
int	maxq
int	qlen
double	utilize
struct pcb_list	*ready
struct pcb_list	*tail

This pcb is not in the ready queue. It is the active pcb!

pcb_list

struct pcb_type	*pcb
struct pcb_list	*next
struct pcb_list	*prev



pcb_type Major Data Structures

char	status
char	flags
char	user[5]
unsigned int	termnl
int	*script
int	pgmid
struct segment_type	*segtable
unsigned int	segtab_len
struct state_type	cpu_save
struct simtime	logont
struct simtime	blockedt
struct simtime	readyt
struct simtime	runt
unsigned long	sjnburst
double	sjnave
unsigned long	instrleft
unsigned long	sliceleft
struct simtime	tblocked
struct simtime	tready
struct simtime	trun
struct simtime	tlogon
int	served
double	efficncy
struct rb_type	*wait
struct rb_list	*firstrb
struct rb_list	*lastrb
struct simtime	waitq
struct simtime	busy

calloc(MAXSCRIPT, sizeof(int))

0	1	2	…	MAXSCRIPT-1

calloc(nsegs, sizeof(struct segment_type))

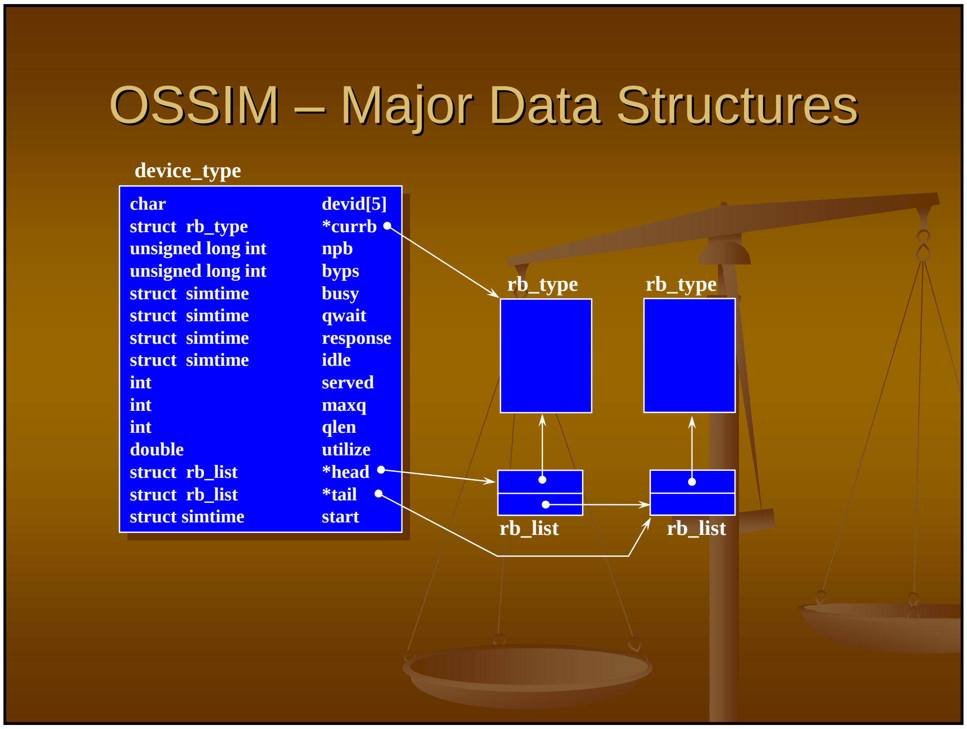
0	1	2	…	nsegs -1

rb_type

char	status
struct pcb_type	*pcb
int	dev
struct simtime	queuet
unsigned long	bytes
struct addr_type	reqid

rb_list

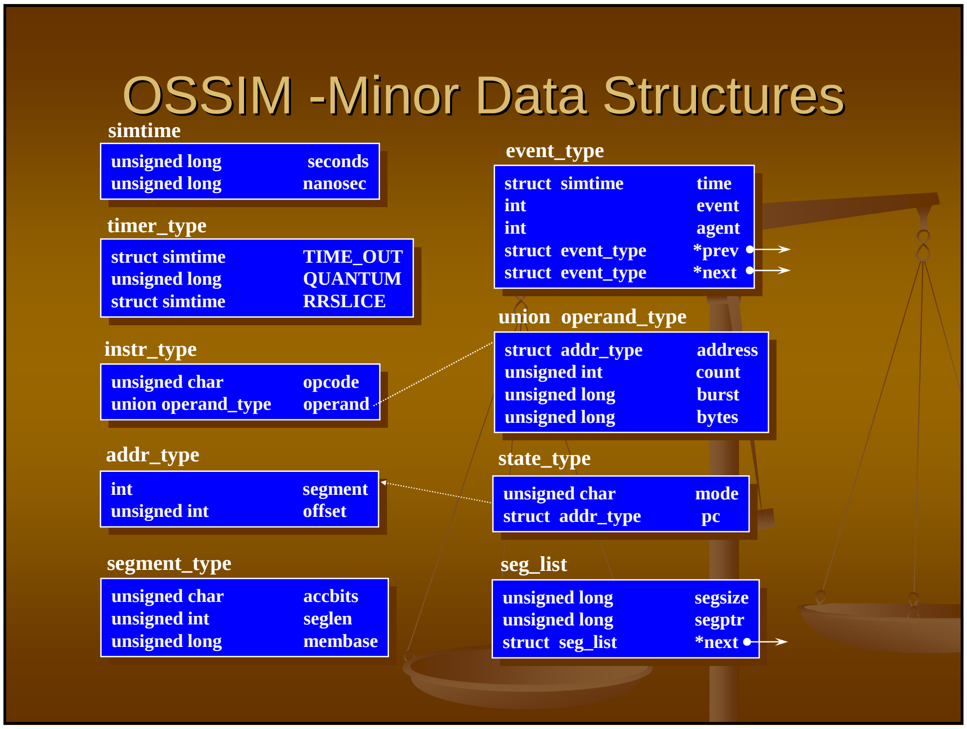
struct rb_type	*rb
struct rb_list	*next



OSSIM – Major Data Structures

device_type

char	devid[5]
struct rb_type	*currb
unsigned long int	npb
unsigned long int	byps
struct simtime	busy
struct simtime	qwait
struct simtime	response
struct simtime	idle
int	served
int	maxq
int	qlen
double	utilize
struct rb_list	*head
struct rb_list	*tail
struct simtime	start



OSSIM -Minor Data Structures

simtime

unsigned long	seconds
unsigned long	nanosec

timer_type

struct simtime	TIME_OUT
unsigned long	QUANTUM
struct simtime	RRSLICE

instr_type

unsigned char	opcode
union operand_type	operand

addr_type

int	segment
unsigned int	offset

segment_type

unsigned char	accbits
unsigned int	seglen
unsigned long	membase

event_type

struct simtime	time
int	event
int	agent
struct event_type	*prev
struct event_type	*next

union operand_type

struct addr_type	address
unsigned int	count
unsigned long	burst
unsigned long	bytes

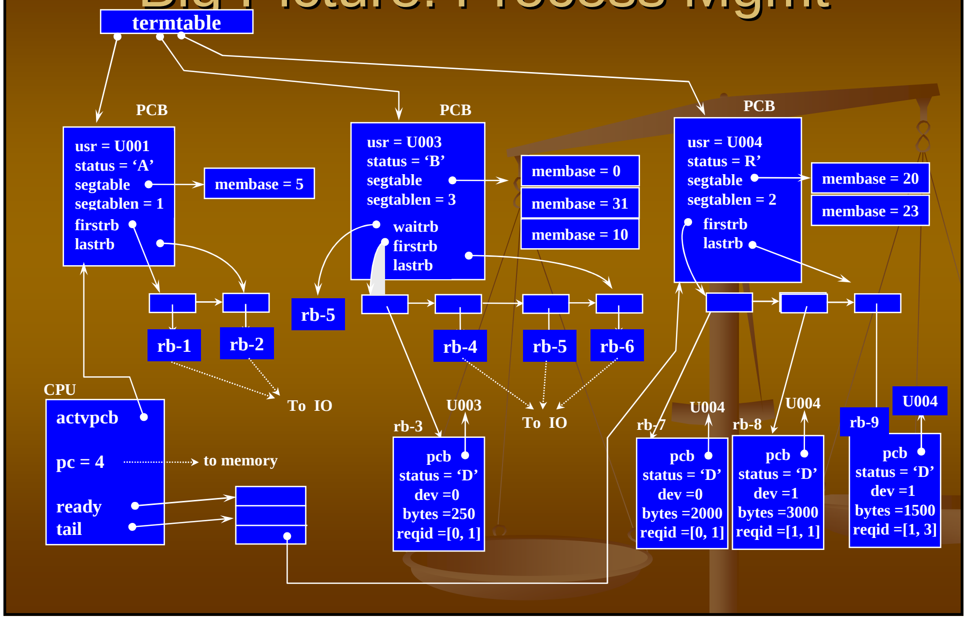
state_type

unsigned char	mode
struct addr_type	pc

seg_list

unsigned long	segsize
unsigned long	segptr
struct seg_list	*next

Big Picture: Process Mgmt



termtable

PCB

usr = U001
status = ‘A’
segtable
segtablen = 1
firstrb
lastrb

membase = 5

PCB

usr = U003
status = ‘B’
segtable
segtablen = 3
waitrb
firstrb
lastrb

membase = 0

membase = 31

membase = 10

PCB

usr = U004
status = R’
segtable
segtablen = 2
firstrb
lastrb

membase = 20

membase = 23

rb-1

rb-2

rb-5

rb-4

rb-5

rb-6

To IO

To IO

CPU

actvpcb

pc = 4

to memory

ready
tail

rb-3

U003

pcb
status = ‘D’
dev =0
bytes =250
reqid =[0, 1]

rb-7

U004

pcb
status = ‘D’
dev =0
bytes =2000
reqid =[0, 1]

rb-8

U004

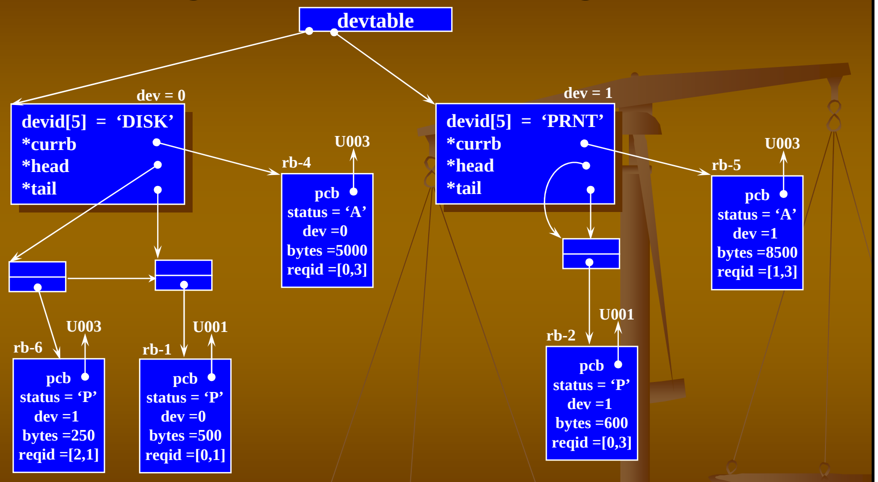
pcb
status = ‘D’
dev =1
bytes =3000
reqid =[1, 1]

rb-9

U004

pcb
status = ‘D’
dev =1
bytes =1500
reqid =[1, 3]

Big Picture: IO Mgmt



devtable

dev = 0

devid[5] = ‘DISK’
*currb
*head
*tail

rb-4

U003

pcb
status = ‘A’
dev =0
bytes =5000
reqid =[0,3]

dev = 1

devid[5] = ‘PRNT’
*currb
*head
*tail

rb-5

U003

pcb
status = ‘A’
dev =1
bytes =8500
reqid =[1,3]

rb-6

U003

pcb
status = ‘P’
dev =1
bytes =250
reqid =[2,1]

rb-1

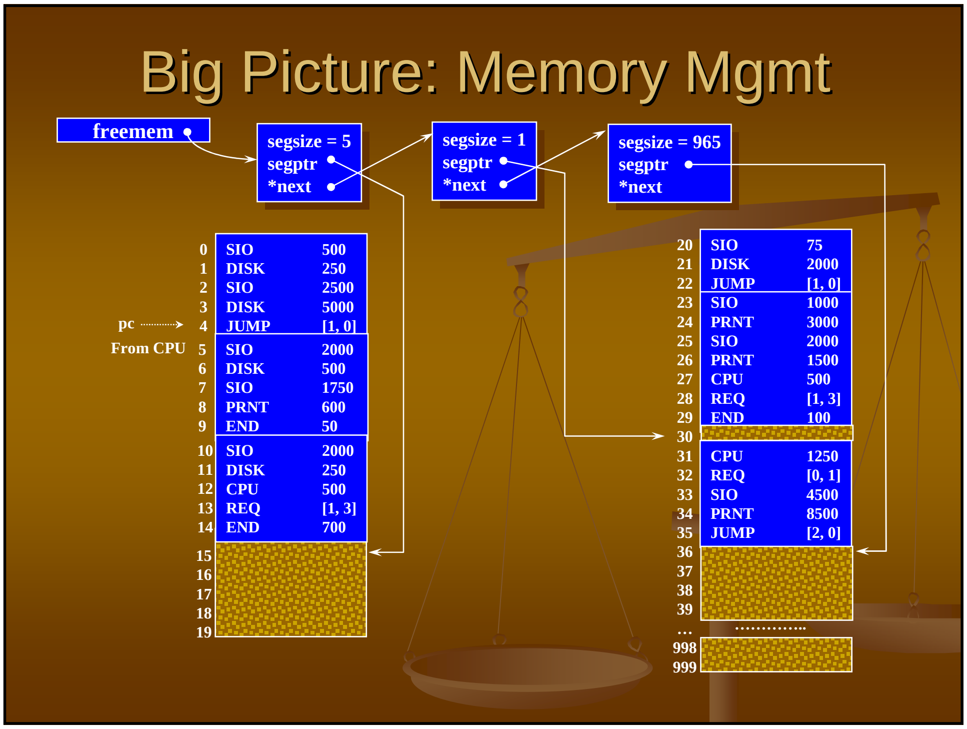
U001

pcb
status = ‘P’
dev =0
bytes =500
reqid =[0,1]

rb-2

U001

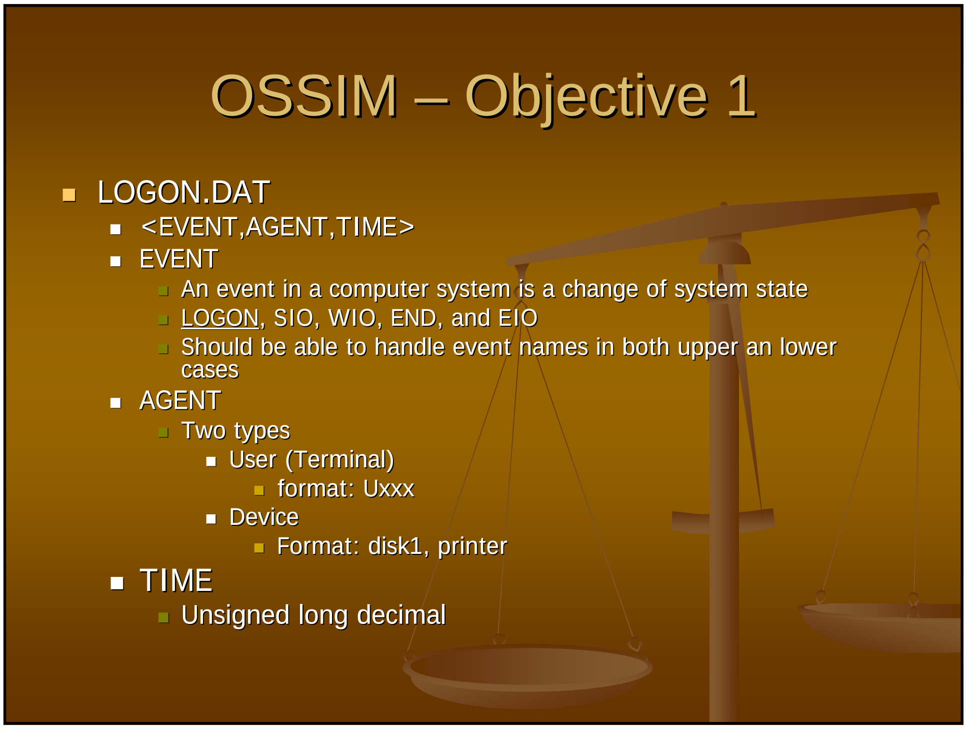
pcb
status = ‘P’
dev =1
bytes =600
reqid =[0,3]



Big Picture: Memory Mgmt

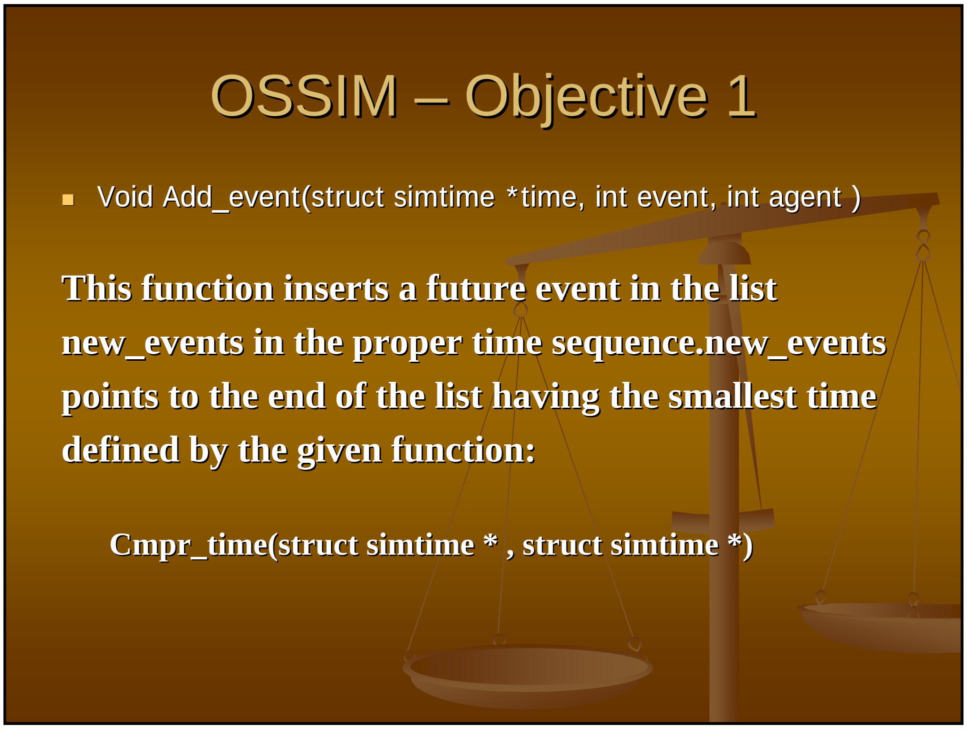
0	SIO	500
1	DISK	250
2	SIO	2500
3	DISK	5000
4	JUMP	[1, 0]
5	SIO	2000
6	DISK	500
7	SIO	1750
8	PRNT	600
9	END	50
10	SIO	2000
11	DISK	250
12	CPU	500
13	REQ	[1, 3]
14	END	700
15		
16		
17		
18		
19		

20	SIO	75
21	DISK	2000
22	JUMP	[1, 0]
23	SIO	1000
24	PRNT	3000
25	SIO	2000
26	PRNT	1500
27	CPU	500
28	REQ	[1, 3]
29	END	100
30		
31	CPU	1250
32	REQ	[0, 1]
33	SIO	4500
34	PRNT	8500
35	JUMP	[2, 0]
36		
37		
38		
39		
…		
998		
999		



OSSIM – Objective 1

- LOGON.DAT
  - <EVENT,AGENT,TIME>
  - EVENT
    - An event in a computer system is a change of system state
    - LOGON, SIO, WIO, END, and EIO
    - Should be able to handle event names in both upper an lower cases
  - AGENT
    - Two types
      - User (Terminal)
        - format: Uxxx
      - Device
        - Format: disk1, printer
  - TIME
    - Unsigned long decimal

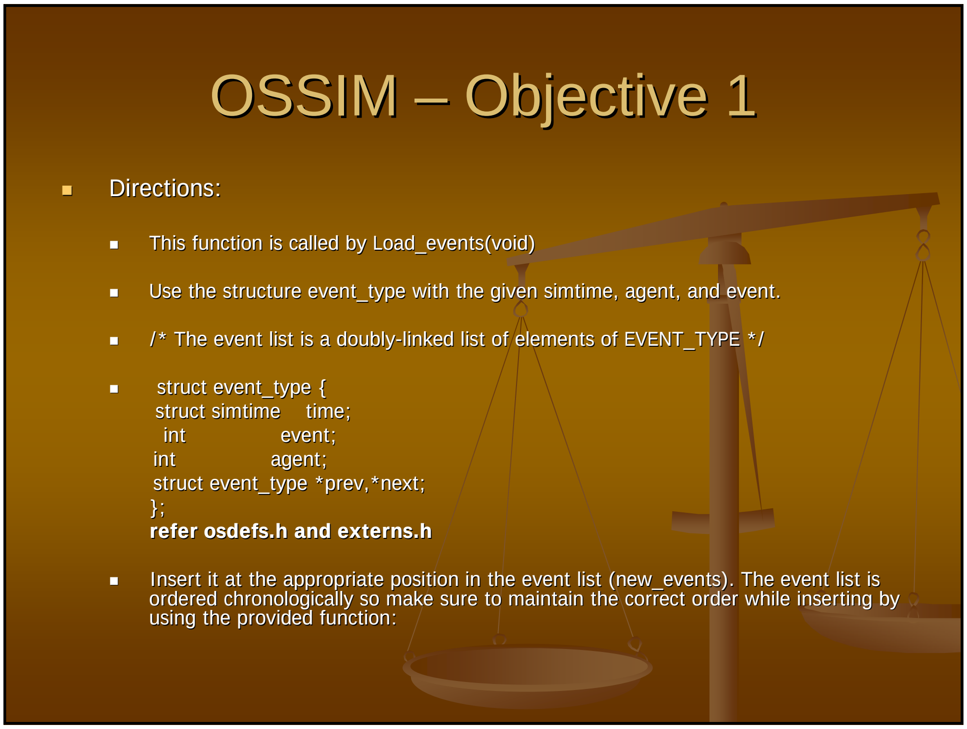


OSSIM – Objective 1

- Void Add_event(struct simtime *time, int event, int agent)

This function inserts a future event in the list new_events in the proper time sequence.new_events points to the end of the list having the smallest time defined by the given function:

Cmpr_time(struct simtime * , struct simtime *)



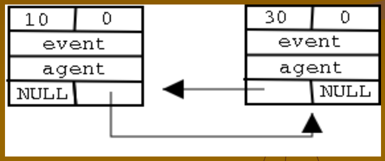
OSSIM – Objective 1

- Directions:
  - This function is called by Load_events(void)
  - Use the structure event_type with the given simtime, agent, and event.
  - /* The event list is a doubly-linked list of elements of EVENT_TYPE */

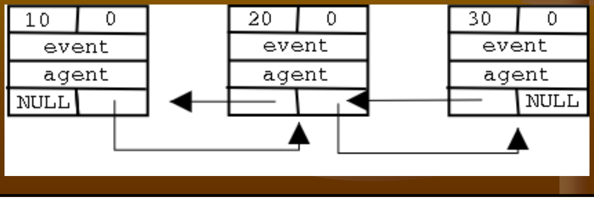
  -
    ```
     struct event_type {
     struct simtime    time;
      int               event;
    int               agent;
    struct event_type *prev,*next;
    };
    ```
    refer osdefs.h and externs.h
  - Insert it at the appropriate position in the event list (new_events). The event list is ordered chronologically so make sure to maintain the correct order while inserting by using the provided function:

OSSIM – Objective 1

- Before:



- After inserting a simtime record with seconds = 20, nanosec = 0

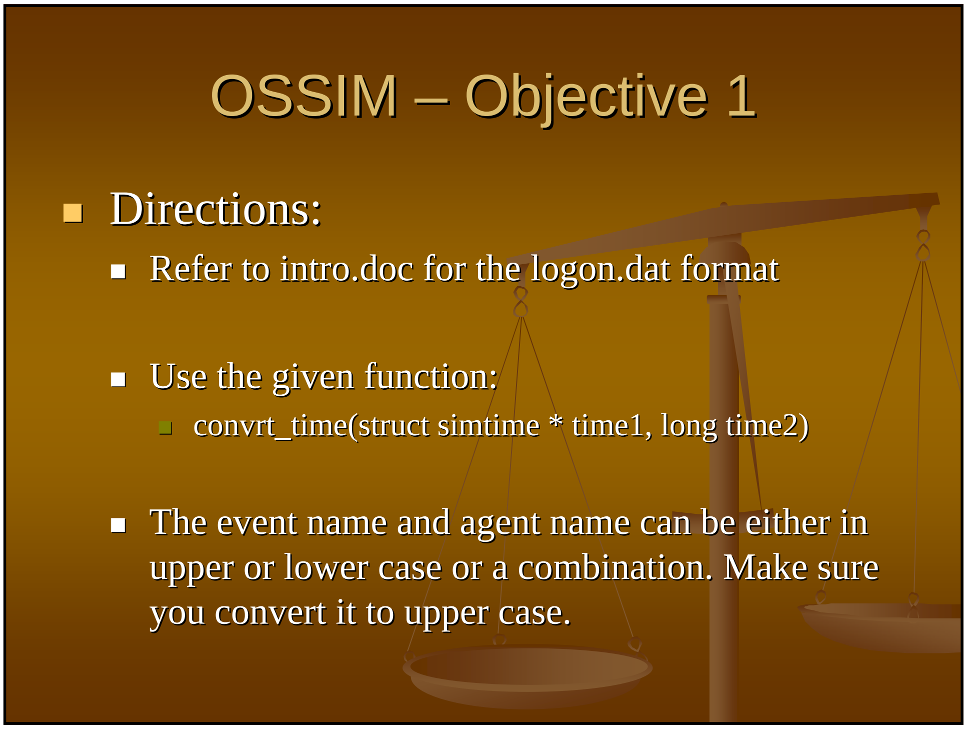


OSSIM – Objective 1

- void Load_events(void)

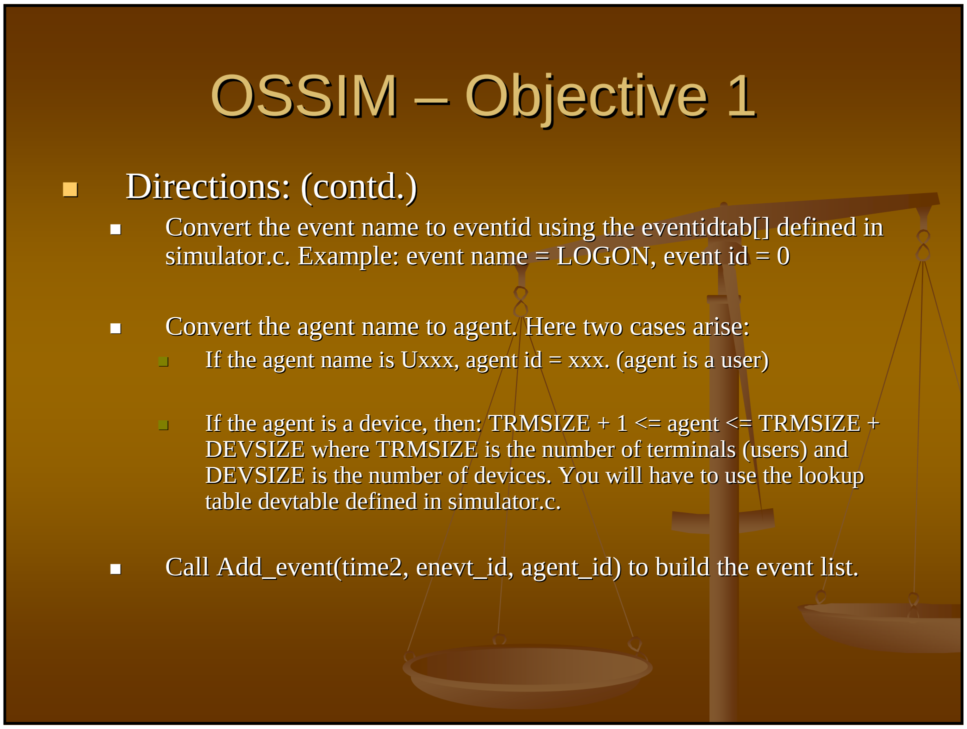
This function is called from simulator.c (The simulator driver) and it initializes the event list (new_events) from the file logon.dat. This file normally contains only LOGON events for all terminals. However, for debugging purposes, logon.dat can contain events of any type. This function uses:

Add_event(struct simtime * , int, int)



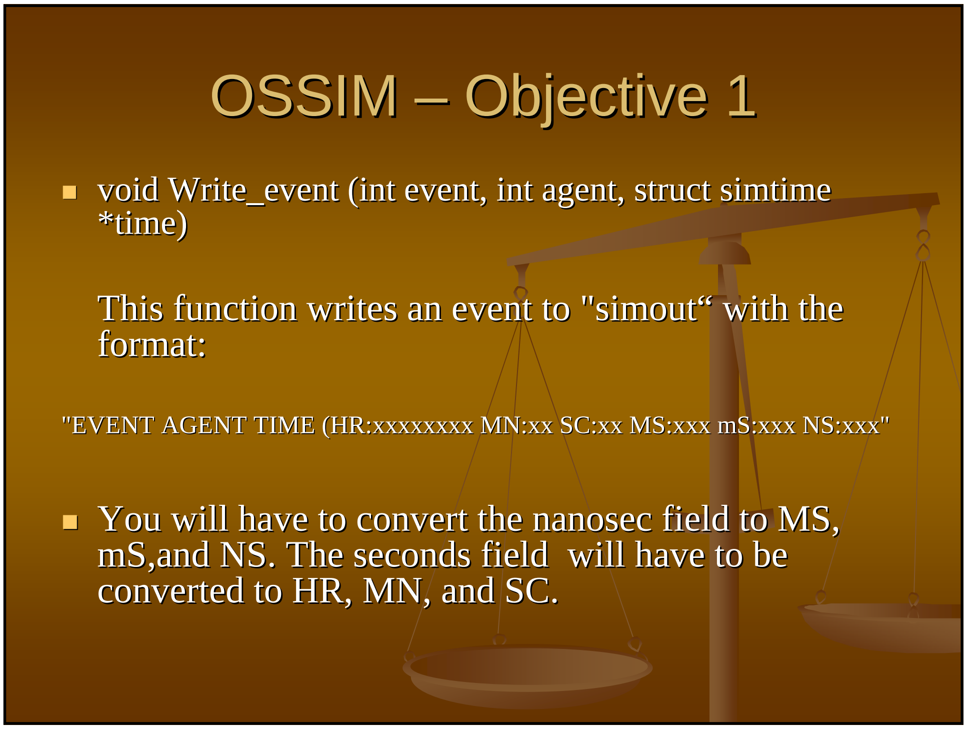
OSSIM – Objective 1

- Directions:
  - Refer to intro.doc for the logon.dat format
  - Use the given function:
    - convrt_time(struct simtime * time1, long time2)
  - The event name and agent name can be either in upper or lower case or a combination. Make sure you convert it to upper case.



OSSIM – Objective 1

- Directions: (contd.)
  - Convert the event name to eventid using the eventidtab[] defined in simulator.c. Example: event name = LOGON, event id = 0
  - Convert the agent name to agent. Here two cases arise:
    - If the agent name is Uxxx, agent id = xxx. (agent is a user)
    - If the agent is a device, then: TRMSIZE + 1 <= agent <= TRMSIZE + DEVSIZE where TRMSIZE is the number of terminals (users) and DEVSIZE is the number of devices. You will have to use the lookup table devtable defined in simulator.c.
  - Call Add_event(time2, enevt_id, agent_id) to build the event list.



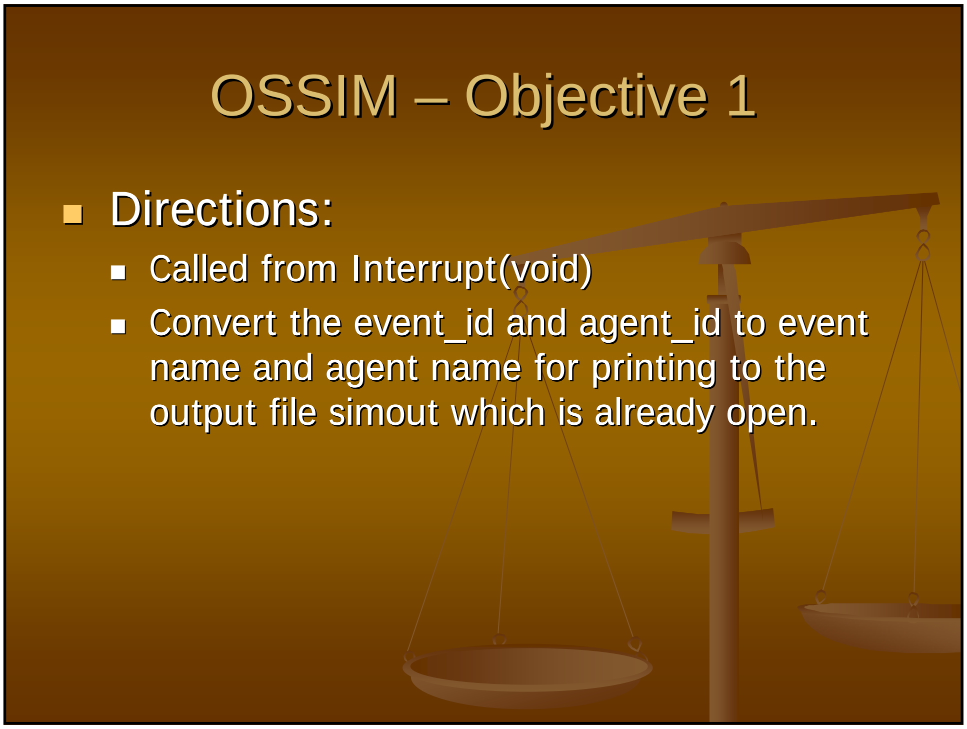
OSSIM – Objective 1

- void Write_event (int event, int agent, struct simtime *time)

  This function writes an event to "simout“ with the format:

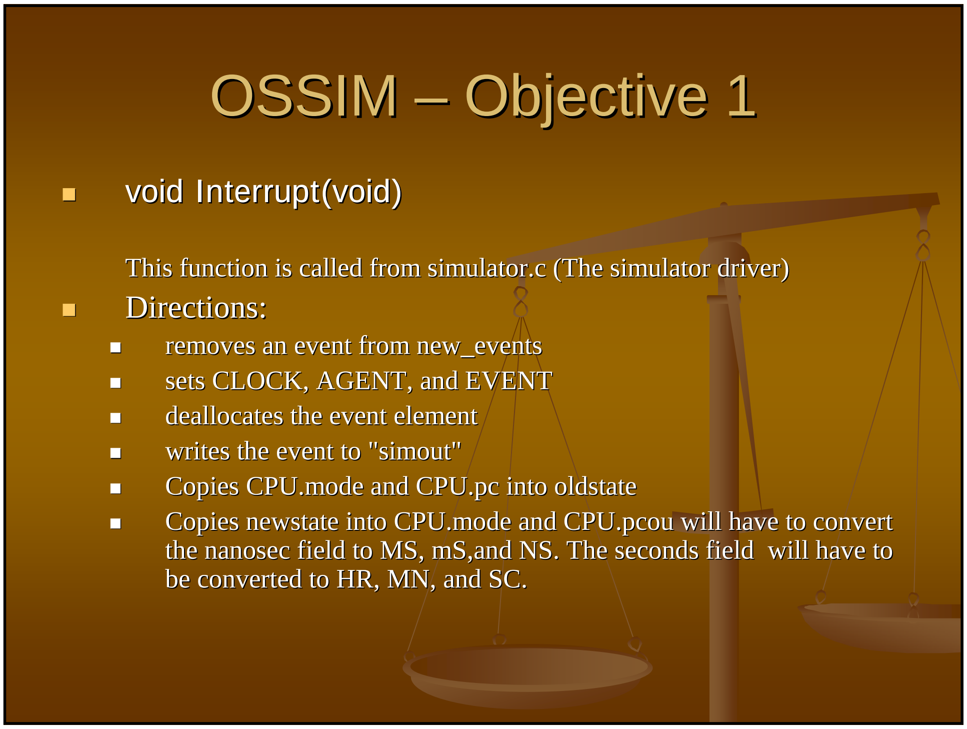
"EVENT AGENT TIME (HR:xxxxxxxx MN:xx SC:xx MS:xxx mS:xxx NS:xxx"

- You will have to convert the nanosec field to MS, mS,and NS. The seconds field will have to be converted to HR, MN, and SC.



OSSIM – Objective 1

- Directions:
  - Called from Interrupt(void)
  - Convert the event_id and agent_id to event name and agent name for printing to the output file simout which is already open.



OSSIM – Objective 1

- void Interrupt(void)

  This function is called from simulator.c (The simulator driver)
- Directions:
  - removes an event from new_events
  - sets CLOCK, AGENT, and EVENT
  - deallocates the event element
  - writes the event to "simout"
  - Copies CPU.mode and CPU.pc into oldstate
  - Copies newstate into CPU.mode and CPU.pcou will have to convert the nanosec field to MS, mS,and NS. The seconds field will have to be converted to HR, MN, and SC.