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 within the OS
    - Fundamental data structures

- **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

- **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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned long</td>
<td>rate</td>
</tr>
<tr>
<td>unsigned long</td>
<td>CPU_burst</td>
</tr>
<tr>
<td>unsigned char</td>
<td>mode</td>
</tr>
<tr>
<td>struct address_type</td>
<td>pc</td>
</tr>
<tr>
<td>struct simtime</td>
<td>*activepcb</td>
</tr>
<tr>
<td>struct simtime</td>
<td>busy</td>
</tr>
<tr>
<td>struct simtime</td>
<td>response</td>
</tr>
<tr>
<td>int</td>
<td>id</td>
</tr>
<tr>
<td>int</td>
<td>served</td>
</tr>
<tr>
<td>double</td>
<td>maxq</td>
</tr>
<tr>
<td>struct pcb_list</td>
<td>qlen</td>
</tr>
<tr>
<td>struct pcb_list</td>
<td>utilize</td>
</tr>
<tr>
<td>*ready</td>
<td></td>
</tr>
<tr>
<td>*tail</td>
<td></td>
</tr>
</tbody>
</table>

#### pcb_type

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>struct pcb_type</td>
<td>*pcb</td>
</tr>
<tr>
<td>struct pcb_list</td>
<td>*next</td>
</tr>
<tr>
<td>struct pcb_list</td>
<td>*prev</td>
</tr>
</tbody>
</table>

#### pcb_list

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>struct pcb_type</td>
<td>pcb</td>
</tr>
<tr>
<td>struct pcb_list</td>
<td>nextpcb</td>
</tr>
<tr>
<td>struct pcb_list</td>
<td>prevpcb</td>
</tr>
</tbody>
</table>

#### pcb_type

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>status</td>
</tr>
<tr>
<td>char</td>
<td>flags</td>
</tr>
<tr>
<td>unsigned int</td>
<td>user[S]</td>
</tr>
<tr>
<td>int</td>
<td>termnl</td>
</tr>
<tr>
<td>int</td>
<td>*script</td>
</tr>
<tr>
<td>int</td>
<td>pgmid</td>
</tr>
<tr>
<td>struct segment_type</td>
<td>*segtab</td>
</tr>
<tr>
<td>unsigned int</td>
<td>segtab_len</td>
</tr>
<tr>
<td>struct state_type</td>
<td>cpu_save</td>
</tr>
<tr>
<td>struct simtime</td>
<td>logont</td>
</tr>
<tr>
<td>struct simtime</td>
<td>blockedt</td>
</tr>
<tr>
<td>struct simtime</td>
<td>readyt</td>
</tr>
<tr>
<td>struct simtime</td>
<td>runt</td>
</tr>
<tr>
<td>unsigned long</td>
<td>sjnburst</td>
</tr>
<tr>
<td>double</td>
<td>sjnave</td>
</tr>
<tr>
<td>unsigned long</td>
<td>instrleft</td>
</tr>
<tr>
<td>double</td>
<td>sliceleft</td>
</tr>
<tr>
<td>unsigned long</td>
<td>blocked</td>
</tr>
<tr>
<td>struct simtime</td>
<td>tready</td>
</tr>
<tr>
<td>struct simtime</td>
<td>trun</td>
</tr>
<tr>
<td>struct simtime</td>
<td>tlogon</td>
</tr>
<tr>
<td>int</td>
<td>served</td>
</tr>
<tr>
<td>double</td>
<td>efficiency</td>
</tr>
<tr>
<td>struct rb_type</td>
<td>*pcbr</td>
</tr>
<tr>
<td>struct rb_type</td>
<td>*nextrb</td>
</tr>
<tr>
<td>struct rb_list</td>
<td>*lastrb</td>
</tr>
<tr>
<td>struct rb_list</td>
<td>*wqrb</td>
</tr>
<tr>
<td>struct rb_list</td>
<td>*busr</td>
</tr>
<tr>
<td>struct simtime</td>
<td>busy</td>
</tr>
</tbody>
</table>

#### rb_type

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>status</td>
</tr>
<tr>
<td>struct pcb_type</td>
<td>*pcb</td>
</tr>
<tr>
<td>int</td>
<td>dev</td>
</tr>
<tr>
<td>struct simtime</td>
<td>waitq</td>
</tr>
<tr>
<td>unsigned long</td>
<td>bytes</td>
</tr>
<tr>
<td>struct addr_type</td>
<td>reqid</td>
</tr>
</tbody>
</table>

#### rb_list

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>struct rb_type</td>
<td>*rb</td>
</tr>
<tr>
<td>struct rb_list</td>
<td>*nextrb</td>
</tr>
<tr>
<td>struct rb_list</td>
<td>*lastrb</td>
</tr>
<tr>
<td>struct rb_list</td>
<td>*wqrb</td>
</tr>
</tbody>
</table>

#### rb_type

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inpcb</td>
<td></td>
</tr>
<tr>
<td>struct rb_type</td>
<td></td>
</tr>
<tr>
<td>struct rb_list</td>
<td>*rb</td>
</tr>
<tr>
<td>struct rb_list</td>
<td>*next</td>
</tr>
<tr>
<td>struct rb_list</td>
<td>*lastrb</td>
</tr>
<tr>
<td>struct simtime</td>
<td>busy</td>
</tr>
</tbody>
</table>
OSSIM – Major Data Structures

device_type

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

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
- unsigned int
  segment
  offset

segment_type

- unsigned char
  accbits
- unsigned int
  seglen
- unsigned long
  membase

event_type

- struct simtime
  time
- int
  event
  agent
- struct event_type
  prev
  next
- struct event_type
  prev
  next
- unsigned char
  mode
- union addr_type
  address
  count
  burst
  bytes
- struct addr_type
  pc
- struct seg_list
  segsize
  segptr
  next

seg_list
**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 and 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

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, event_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:xxxxxxx 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.