COSC 3360/6310 - Operating Systems Fall 2015

Programming Assignment 2: Unix/Linux Semaphores and EDF Scheduling

Due Date: Wednesday, November 11, 2015, 11:59pm

In this assignment, you will implement a program to handle hotel reservations requested by customers at different locations connected to the "Fall-OS" hotel's database.

Suppose each customer is represented by a process and the body of the process consists of all the reservations made by that customer. A customer can make reservations, cancel reservations, check (show) reservations, and pay for reserved rooms. Reservations for rooms can be canceled after the reservations have been made by the same customer, but paid rooms cannot be canceled.

Two or more customers may be doing the same transactions at around the same time. Obviously, reservations/payments to the same room/time period must be performed atomically (mutual exclusively). If a customer tries to reserve a room that has already been taken, then disallow this action by skipping it and print a "room taken" message. If a transaction tries to operate a non-existent room, print an error message. A customer can reserve a contiguous or non-contiguous block of two or more rooms at the same time.

To ensure these concurrent operations yield correct results, you are to use Unix semaphores to control access to rooms/stay periods, which are stored in shared variables.

To simulate (1) the transmission delay between the hotel's central computer and a customer's computer, and (2) the processing of each transaction, the first four lines in the body of each customer specify the required total execution time (in milliseconds) for each of the three operations performed at that customer. In your implementation, each specified time is the length of the critical section for the corresponding transaction.

Valid transactions are:

```
reserve (room_number) begin_date number_of_days name_of_customer deadline d1
cancel (room_number) begin_date number_oF_days name_of_customer deadline d2
reserve (room_number1,...,room_numberK) begin_date number_of_days name_of_customer deadline d3
      /* non-contiguous block */
cancel (room_number1,...,room_numberK) begin_date number_of_days name_of_customer deadline d4
reserve (room_number1-room_numberM) begin_date number_of_days name_of_customer deadline d5
      /* contiguous block */
cancel (room_number1-room_numberM) begin_date number_of_days name_of_customer deadline d6
check customer_name deadline d7  /* show rooms and stay periods reserved */
pay (room_number) begin_date number_of_days name_of_customer deadline d8
```

Example–reserving a non-contiguous block of rooms:
reserve (2,5,58) ...
Example–reserving a contiguous block of rooms:
reserve (8-15) ...
You can cancel a room reservation only after you have reserved it.

Each transaction is followed by the keyword deadline and its numerical value (relative to the start time of the process or process creation time) for completing this transaction.

The input is as follows:

```
n     /*  number of rooms, numbered from 1 to n  */
m     /* number of customers */
customer_1:
reserve reserve_time (in milliseconds)
cancel cancel_time   (in milliseconds)
check check_time (in milliseconds)
pay pay_time (in milliseconds)
  :
valid operations
  :
end.

  :
  :
  :

customer_m:
reserve reserve_time (in milliseconds)
cancel cancel_time   (in milliseconds)
check check_time (in milliseconds)
pay pay_time (in milliseconds)
  :
valid operations
  :
end.
```

Implement two versions: (1) one without considering customers' deadlines; and (2) another scheduling (using EDF) the transactions according to their deadlines.

The output for each implementation after completing all transactions is: a report showing the transactions and resulting room assignments with customer names, and whether each transaction's deadline is met for every customer (if not, show the lateness in milliseconds).