Práctica final: Hotel Palace Sokovia

Miguel Ángel Conde González
Antonio Gómez García
Mario Enrique Casado García

17 de diciembre de 2021

Definición del problema

El objetivo de la práctica es realizar un programa que simule la atención de la recepción del Hotel Palace Sokovia. Para ello los recepcionistas de diferentes tipos y los clientes se representarán mediante threads. La práctica tiene una parte básica y que es obligatorio implementar (supondrá como mucho el 80 % de la nota) y, además, se proponen una serie de mejoras opcionales para aumentar la nota (como mucho supondrá el 20 % de la nota).

Para aprobar la práctica es necesario que la parte básica funcione correctamente. La nota se asignará en base a la calidad del código entregado, valorándose:

- Política de nombres (coherencia en el nombrado de variables y funciones).
- Sangrado (indentación).
- Comentarios (cantidad, calidad y presentación).
- Legibilidad: Nombre de variables, funciones, etc.
- Reusabilidad y mantenibilidad: Uso de parámetros, etc.

La práctica es en grupo y se evaluará según la metodología CTMTC explicada en clase.

Parte Básica (80 % de la nota)

La aplicación funciona de la siguiente manera:

- Los clientes van a entrar en el hotel y podrán ser de dos tipos, clientes normales y clientes vip.
- Debidos a las restricciones COVID19 a la recepción solamente van a poder entrar 20 personas a la vez, si se supera esa cifra la persona no podrá entrar y se marchará. Para atender a los clientes se cuenta con 3 recepcionistas y 5 máquinas de autocheckin. Dos de los recepcionistas atenderán a clientes normales mientras que otro atenderá a los clientes vip. Por otro lado las máquinas prestan atención pero no están vinculadas a una persona.

- Un 10 % de los clientes (tanto normales como vip) decide ir directamente a la máquina atendedora en lugar de a recepción
- Un 20 % de los clientes, se cansa de esperar y se va a las máquinas automáticas y otro 10 % se cansa de esperar y abandona el hotel. Del 70 % restante un 5 % pierde el turno por ir al baño mientras espera y abandona el hotel.
- Si un cliente va a la zona de checkin automático y tiene que esperar en un 50 % de ocasiones cambiará de idea y se irá a la recepción normal.
- A cada cliente se le asignará un identificador único y secuencial (cliente_1, cliente_2,... cliente_N) a medida que vayan entrando en el hotel.
- Una vez un cliente ha sido atendido, podrá esperar por el ascensor esto les pasa a un 30 % el resto comunica que se va a su habitación y termina. Los que van al ascensor intentan usar el ascensor y permanecen en él sistema hasta que lo hacen.

Aspectos a tener en cuenta:

- Los recepcionistas tienen un un identificador único (recepcionista_1, recepcionista_2, recepcionista_3).
- Solo cuando se haya atendido a un cliente éste pasará a intentar entrar en el ascensor o se marchará.
- Los clientes comprueban cada 3 segundos si se marchan por alguna de las posibles razones (se cansan y se van, se cansan y pasan a las máquinas de autocheckin o van al baño y pierden turno) y en caso de irse abandonan la cola y de no ser así esperarían procederían con acción.
- Cada vez que se atienden 5 clientes, el recepcionista descansa 5 segundos, con lo que o los atiende otro compañero o tendría que atenderlos.
- El recepcionista vip nunca descansa, pero solamente va a atender a clientes vip.
- Los clientes de cada tipo deben atenderse por orden de llegada.
- Si un cliente trata de acceder a las máquinas de autocheckin y no hay ninguna libre tendrá que esperar 3 segundos y volverlo a intentar, en un 50 % de los casos puede tratar de ser atendido de nuevo por un recepcionista.
- En la máquina de autocheckin el cliente se atiende a sí mismo y para simularlo tendrá que esperar 6 segundos antes de pasar a ascensores o abandonar el sistema.
- Si un cliente accede a una máquina de autocheckin no abandona el sistema hasta que no haya sido atendido, con lo que sigue utilizando un hueco de la cola.
- La subida en el ascensor se simula con entre 3 y 6 segundos de espera.
- Si un cliente no puede acceder al ascensor porque éste está funcionando, debe esperar 3 segundos y volver a comprobarlo.

- El ascensor tiene una capacidad de 6 personas y no funcionará hasta que este lleno.
- El último cliente en entrar en el ascensor cierra la puerta y lo pone a funcionar, siendo además el primero en salir.
- Hasta que no se vacía el ascensor no se considera que ha dejado de funcionar.
- De los clientes a atender, el 80 % tiene todo en regla, el 10 % no se ha identificado correctamente y el 10 % no tiene el pasaporte vacunal. Esto tiene una implicación en los tiempos de atención:
  - 80 % todo en regla – En estos casos, el tiempo de espera está entre 1 y 4 segundos y después se pasará o no a los ascensores.
  - 10 % mal identificados – En estos casos, el tiempo de espera está entre 2 y 6 segundos y después se pasar o no a los ascensores.
  - 10 % que no tiene el pasaporte vacunal – En estos casos, el tiempo de espera está entre 6 y 10 segundos y abandonarían el hotel.
- En los dos primeros casos se va a los ascensores según los porcentajes previamente comentados en el último transcurrido el tiempo considerado se abandonará el hotel.

Toda la actividad quedará registrada en un fichero plano de texto llamado `registroTiempos.log`. En concreto, es necesario registrar al menos:

- Cada vez que un cliente accede al hotel
- Cada vez que una cliente deja el hotel por el motivo que sea
- Cada vez que un cliente se hospeda correctamente.
- Cuando a un cliente sube en el ascensor.
- Cuando un cliente va al ascensor
- Se registra el inicio y final del descanso de cada recepcionista.

Consideraciones prácticas:

- Simularemos el inicio de funcionamiento del sistema mediante señales. En caso de que un cliente normal que quiera acceder al hotel se usará la señal SIGUSR1, en el caso de un cliente vip SIGUSR2. Cada vez que se le envíe la señal, supone que un nuevo cliente trata de acceder al consultorio, las señales se envían desde consola.
- Es obligatorio el uso de mensajes que se escribirán en un log y se mostrarán por pantalla. El formato de tales mensajes será:

  `[YYYY-MM-DD HH:MI:SS] identificador: mensaje`

  Donde `identificador` puede ser el identificador del cliente, el identificador del recepcionista o si ha sido atendido por una máquina y `mensaje` es una breve descripción del evento ocurrido.

- Las entradas del log deben quedar escritas en orden cronológico.
- El programa finaliza cuando recibe la señal SIGINT y deberá hacerlo correctamente.
- Al finalizar el programa se debe terminar de atender a todos los clientes en cola, pero ya no podrán subir en el ascensor.

Partes opcionales (20 % de la nota)

- Asignación estática de recursos (10 %):
  - Modifica el programa para que el número de clientes que pueden tratarse en el hotel sea un parámetro que reciba el programa al ser ejecutado desde la línea de comandos.
  - Modifica el programa para que el número de máquinas de autocheckin que atienden clientes sea un parámetro que reciba el programa al ser ejecutado desde la línea de comandos.
- Asignación dinámica de recursos I (5 %):
  - Modifica el programa para que el número de clientes que pueden acceder al hotel pueda modificarse en tiempo de ejecución.
  - Solamente es necesario contemplar un incremento en los clientes. No es necesario contemplar la reducción.
  - Cada vez que se cambie el número de clientes tiene que reflejarse en el log.
- Asignación dinámica de recursos II (5 %):
  - Modifica el programa para que el número de máquinas de autocheckin se pueda modificar en tiempo de ejecución.
  - Solamente es necesario contemplar un incremento en el número de máquinas de autochekin. No es necesario contemplar la reducción.
  - Cada vez que se produce un cambio en este sentido debe quedar reflejado en el log.

Escritura de mensajes en log

Es recomendable utilizar una función parecida a esta para evitar repetir líneas de código. Recibe como parámetros dos cadenas de caracteres, una para el identificador y otra para el mensaje (la fecha la calcula la propia función):

```c
void writeLogMessage(char *id, char *msg) {
  // Calculamos la hora actual
  time_t now = time(0);
  struct tm *tlocal = localtime(&now);
  char stnow[25];
  strftime(stnow, 25, "%d/%m/%y %H:%M:%S", tlocal);

  // Escribimos en el log
  logFile = fopen(logFileName, "a");
  fprintf(logFile, "[%s] %s: %s\n", stnow, id, msg);
  fclose(logFile);
}
```

Ejemplo 1: Diseño de la parte básica