DISEÑO DE LA PRÁCTICA FINAL

Declaraciones GLOBALES: Lo escrito en azul va a fichero, lo rojo a cola, etc

- Semáforos y variables condición
  - Fichero, colaClientes, solicitudes
  - Condiciones para el técnico de atención domiciliaria y los clientes que la solicitan
- Contador de clientes de tipo App
- Contador de clientes de tipo Red
- Número de solicitudes domiciliarias
- Lista de 20 clientes:
  - Id
  - Atendido0 cuando no lo está, 1 cuando se está atendiendo, 2 cuando está atendido
  - Tipo
  - Prioridad
  - Solicitud 0 o 1, si está en momento de solicitud o no
- Técnicos App y Red (lista o no) Si se hace la opcional, los app tienen que ser un puntero
- Fichero de log (FILE * logFile); Reiniciarlo cada vez que se ejecute el programa

main {

1. signal o sigaction SIGUSR1, cliente app.
2. signal o sigaction SIGUSR2, cliente red.
3. signal o sigaction SIGINT, terminar.
4. Inicializar recursos (¡Ojo!, Inicializar!=Declarar).
   a. Semáforos.
   b. Contador de clientes de cada tipo.
   c. Lista de clientes id 0, Prioridad 0, atendido 0, solicitud 0.
   d. Lista de técnicos (si se incluye).
   e. Fichero de Log
   f. Variables relativas a la solicitud de atención domiciliaria.
   g. Variables condición
5. Crear 6 hilos (técnicos, responsables de reparaciones, encargado y atención domiciliaria).
6. Esperar por señales de forma infinita.

}

nuevoClienteRed{ignorar lo de red, nuevoCliente

1. Comprobar si hay espacio en la lista de clientes. Esta parte roja tendría que hacerse dentro del mutex
   a. Si lo hay
      i. Se añade el cliente.
      ii. Contador de clientes se incrementa.
      iii. nuevoCliente.id = ContadorClientesTipo
      iv. nuevoCliente.atendido=0
      v. nuevoCliente.tipo=tipo
      vi. nuevoCliente.solicitud=0.
      vii. nuevoCliente.prioridad=prioridadCalculada
      viii. Creamos hilo para el cliente. Despues de esto se cierra el mutex
   b. Si no hay espacio
      i. Se ignora la llamada. puede imprimirse un mensaje de que ha ignorado por falta de espacio

}

AccionesCliente{

1. Guardar en el log la hora de entrada.
2. Guardar en el log el tipo de cliente.
3. Comprueba si está siendo atendido.

a. Si no lo está, calculamos el comportamiento del cliente (si se va por dificultad o si se cansa de esperar algo que solo ocurre cada 8 segundos) y también el caso de que pierda la conexión a internet.
b. Si se fuera y se escribe en el log, se daría fin al hilo Cliente y se liberaría el espacio en la cola de clientes App.
c. Sino debe dormir 2 segundos y vuelve a 3. vuelve al paso 3

4. Si está siendo atendido por el técnico correspondiente debemos esperar a que termine (puede comprobar cada 2 segundos). si la variable atendido es 1, está siendo atendido.
5. Si el cliente es de tipo red y quiere realizar una solicitud domiciliaria
    a. Comprueba el número de solicitudes pendientes
    b. Si es menor de 4 incrementa el valor
        i. Escribe en el log que espera para ser atendido
        ii. Cambia el valor de solicitud a 1
        iii. Si es el cuarto avisa al técnico con condition_signal
        iv. Se bloquea (condition_wait) hasta que la solicitud pase a ser 0 y por tanto ya se haya finalizado la atención
        v. Comunica que su atención se ha finalizado
    c. Sino duerme 3 segundos y vuelve a A)
6. Sino libera su posición en cola de clientes y se va.
7. Escribe en el log
8. Fin del hilo Cliente.

}

AccionesTécnico{

1. Buscar al cliente para atender de su tipo, atendiendo a la prioridad y sino al que más tiempo lleve esperando.
    a. Si no hay clientes para atender espero un segundo y vuelvo a 1.
2. Cambiamos el flag de atendido. Del cliente al que se pone a atender
3. Calculamos el tipo de atención y en función de esto el tiempo de atención (el 80%, 10%, 10%).
4. Guardamos en el log que comienza la atención
5. Dormimos el tiempo de atención.
6. Guardamos en el log que finaliza la atención
7. Guardamos en el log el motivo del fin de la atención.
8. Cambiamos el flag de atendido Se cambia a 2, corresponde a que ha sido atendido
9. Mira si le toca descansar.
10. Volvemos al paso 1 y buscamos el siguiente.

}

AccionesEncargado{

1. Buscar al cliente para atender primero de red y si no hubiera de tipo app, luego atiende a la prioridad y sino al que más tiempo lleve esperando.
    a. Si no hay clientes para atender espera 3 segundos y vuelvo a 1
2. Cambiamos el flag de atendido.
3. Calculamos el tipo de atención y en función de esto el tiempo de atención (el 80%, 10%, 10%).
4. Guardamos en el log que comienza la atención
5. Dormimos el tiempo de atención.
6. Guardamos en el log que finaliza la atención
7. Guardamos en el log el motivo del fin de la atención.
8. Cambiamos el flag de atendido
9. Volvemos al paso 1 y buscamos el siguiente.

}

AccionesTecnicoDomiciliario{

1. Comprueba el número de solicitudes y se queda bloqueado (cond_wait) mientras sea menor de 4.
2. Guardamos en el log que comienza la atención
3. Duerme un segundo para cada petición
4. Escribe que ha atendido uno.
5. Cambia el valor del flag de solicitud a 0 en el que ha atendido
6. Cuando el último se ha atendido se pone el número de solicitudes a 0
7. Guardamos en el log que se ha finalizado la atención domiciliaria
8. Se avisa a los que esperaban por la solicitud domiciliaria que se ha finalizado (cond_signal).
9. Vuelve a 1.

}

Notas: las zonas coloreadas en rojo, azul y morado son zonas de exclusión mutua que deben ser controladas. Para la parte morada se deben usar variables condición.

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 de vehículo o mecánico y otra para el mensaje (la fecha la calcula la propia función:

```
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);
}
```