DISEÑO DE LA PRÁCTICA FINAL

Declaraciones GLOBALES:

- Semáforos y variables condición
  - Fichero, colaSolicitudes, colaSocial
  - Condiciones para el coordinador y los participantes en la actividad social
- Contador de solicitudes
- Lista de 15 usuarios:
  - Id
  - Atendido
  - Tipo
- Lista de 4 usuarios en actividad social
- Atendedores (lista o no)
- Fichero de log (FILE * logFile);

main {

1. signal o sigaction SIGUSR1, solicitud por invitacion
2. signal o sigaction SIGUSR2, solicitud por QR
3. signal o sigaction SIGINT, terminar
4. Inicializar recursos (¡Ojo!, Inicializar!=Declarar).
   a. Semáforos.
   b. Contador de solicitudes.
   c. Lista de solicitudes id 0, atendido 0, facturado 0.
   d. Lista de atendedores (si se incluye).
   e. Lista de solicitudes para actividades sociales
   f. Fichero de Log
   g. Variables condición
5. Crear 3 hilos atendedores.
6. Crear el hilo coordinador
7. Esperar señal SIGUSR1, SIGUSR2 o SIGINT
8. Esperar por señales de forma infinita.

}

nuevaSolicitud{

1. Comprobar si hay espacio en la lista de solicitudes.
   a. Si lo hay
      i. Se añade la solicitud.
      ii. Contador de solicitudes se incrementa.
      iii. nuevaSolicitud.id = ContadorSolicitudes.
      iv. nuevaSolicitud.atendido=0
      v. tipo=Depende de la señal recibida.
      vi. Creamos hilo para la solicitud.
   b. Si no hay espacio
      i. Se ignora la llamada.

}

AccionesSolicitud{

1. Guardar en el log la hora de entrada.
2. Guardar en el log el tipo de solicitud.
3. Duerme 4 segundos
4. Comprueba si está siendo atendido.
5. Si no lo está, calculamos el comportamiento de la solicitud (si se va por no ser fiable o por cansarse de esperar QR) o si le afecta lo mal que funciona la invitación.

    a. Si se fuera y se escribe en el log, se daría fin al hilo Usuario y se liberaría el espacio en la cola.
    b. Sino debe dormir 4 segundos y vuelve a 4.
6. Si está siendo atendido por el usuario debemos esperar a que termine.
7. Cuando termine calculamos si decide o no participar en una actividad social
8. Si decide participar
    a. Si puede entrar en la lista de la actividad
        i. Entrar en la lista
        ii. Si es el último debe avisar al coordinador
        iii. Liberar espacio en la cola de solicitudes
        iv. Guardamos el log en que está preparado para la actividad
        v. Se queda esperando a que digan que podemos comenzar
        vi. Duerme 3 segundos
        vii. Sale de la cola y el último avisa al coordinador
        viii. Guardamos el log en que deja la actividad
    b. Sino puede
    c. Espera 3 segundos y vuelve a A)
9. Si no decide participar
    a. Libera su posición en cola de solicitudes y se va
    b. Escribe en el log
10. Fin del hilo Usuario.

}

AccionesAtendedor{

1. Buscar la primera solicitud para atender de su tipo, esto es el que más tiempo lleve esperando**.
    a. Si no hay de mi tipo busco uno del otro.
    b. Si no hay usuarios para atender espero un segundo y vuelvo a 1.
2. Calculamos el tipo de atención y en función de esto el tiempo de atención (el 70%, 20%, 10%).
3. Cambiamos el flag de atendido.
4. Guardamos en el log la hora de atención.
5. Dormimos el tiempo de atención.
6. Guardamos en el log la hora de fin de la atención.
7. Guardamos en el log el motivo del fin de la atención.
8. Cambiamos el flag de atendido.
9. Mira si le toca tomar café.
10. Volvemos al paso 1 y buscamos el siguiente (siempre con prioridad a su tipo).

}

** Si se tata de un atendedor pro cogerá la solicitud que más tiempo lleve independientemente del tipo

Acciones Coordinador Social{

1. Espera que le avisen de que hay 4 y cierra la lista para que ninguno más entre.
2. Avisa a los procesos de que comiencen la actividad
3. Escribe en el log que comienza la actividad
4. Espera que le digan que han finalizado
5. Escribe en el log que finaliza la actividad
6. Abre la lista de nuevo y vuelve a 1

}

Notas: las zonas coloreadas en rojo, azul y morado son zonas de exclusión mutua que deben ser controladas. Para la parte verde 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[19];
       strftime(stnow, 19, "%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);
}
```