## Grado de Ingegneria en informatica

Programacion Multi-Hilo

# Sistemas Operativos

### Pratica 3

Davide Mercanti [100390177@alumnos.uc3m.es](mailto:100390177@alumnos.uc3m.es)

Lorenzo Melis [100390168@alumnos.uc3m.es](mailto:100390168@alumnos.uc3m.es)

# INDICE

Nota Inicial pag 2

Description del codigo pag 3

Bateria de Pruebas pag 4

Comentarios pag 5

**Nota Inicial**

*Esta memoria fue escrita de la mano de dos estudiantes Erasmus italianos, entonces nos disculpamos para todo los errores de gramática que habrá.*

# **Description del codigo**

En general

El codigo trata basicamente de crear tres hilos que tienen roles diferentes y que comparten el mismo buffer de memoria:

* un Jefe de Pista que produce n aviones que despuegan;
* un radar que produce m aviones que atterizan;
* una torre de control que “consuma” los n+m aviones.

Entonces, es basicamente un problema productor/consumidor con dos productores y un consumidor que comparten el dicho buffer.

Arcport

Por primero vamos a describir el contenido del fichero arcport.c.

Aquí tenemos que crear los tres hilos con los tres roles mencionados.  
En el main encontramos por primero un control de los argumentos y la llamada a *queue\_init*, luego la creacion de los hilos con la utilizacion de *pthread\_create.* Despues esperamos por la terminacion de los threads *pthread\_join* y al final liberemos la memoria.

Los hilos creatos ejecutan las siguientes funciones:

* *jefe\_de\_pista*

Maneja los aviones que deben despuegar. Por primero asigna la memoria para la estructura *plane* y pasa el apuntador a *queue\_put* (las funciones de gestion del bufer solo trabajan con apuntadores). Los aviones se almacenan en el heap entonces siguen existendo hasta el free corespondiente.

* *radar*

Hace lo mismo con los aviones que deben atterizar.

* *torre\_de\_control*

Empieza llamando *a queue\_get* para obtener un avion y simplemente ejecuta sleep para simular la accion del avion. Esto se hace en un while; cuando se detecta que el avion es el ultimo el thread termina.

Queue

El fichero *queue.c* crea y maneja el buffer; tienes seis funciones para las varias tareas:

* *queue\_init*

Asigna la memoria para el bufer y inicializa los mutex.

* *queue\_put*

Pone en el bufer el apuntador pasado como parametro. Esta operacion (seccion critica) está protegida por el mutex.

* *queue\_get*

Retorna el primero apuntador listo en el bufer. Esta operacion también (seccion critica) está protegida por el mutex.

* *queue\_empty*

Retorna la negación del valor de la expresion logica ( n\_elementos ), donde n\_elementos es una variable actualizada por las funciones anteriores.

* *queue\_full*

De forma similar retorna el valor de ( n\_elementos==buffer\_size ).

* *queue\_destroy*

Deasigna la memoria del bufer y de los mutexes.

# Funcionalidad y baterias de Pruebas

### Function base:

El funcionamento de base es muy simple, de hecho tenemos 2 modalidad de ejecucion:

* El primero solo llama *arcport* y usa parametros por defecto.
* El segundo llama *arcport* mas 5 argumentos que deben ser interos.

En ambos modos, debido a que los hilos se ejecutan en paralelo, el orden de los mensajes puede cambiar. La exclusion mutua de las operaciones de gestion de la cola circular es garantizada por los mutex, aquella de las gestion de la pista es garantizada tambien por el hecho de que sleep() bloquea la torre de control, la cual es la única que actúa sobre la pista.

### Casos Extremos:

**NB:** La gestion de los errores (e.g. tamaño nulo) es descrita aparte abajo.

* **Tamaño del bufer:** ………
* **Tiempo de action de los aviones:** ……...

Errores:

# Conclusiones

La parte de major dificultade fue la sincronizacion de los hilos con los problemas de concurrencia