Javier García Borrego

Practica 1 - Minikernel

Ampliación de sistemas operativos

Doble grado en Ing. Informática + Ing. Computadores

Contenido

[Llamada que bloquea al proceso un plazo de tiempo: 1](#_Toc106055582)

[Mutex: 1](#_Toc106055583)

[Función Crear\_mutex: 1](#_Toc106055584)

[Función abrir\_mutex: 2](#_Toc106055585)

[Función lock: 2](#_Toc106055586)

[Función unlock: 2](#_Toc106055587)

[Función Cerrar\_mutex: 2](#_Toc106055588)

[Funciones auxiliares: 2](#_Toc106055589)

[Round Robin: 3](#_Toc106055590)

[Ejemplos de ejecución: 3](#_Toc106055591)

[DORMIR: 3](#_Toc106055592)

[Prueba mutex 1 y prueba mutex 2 resultados: 5](#_Toc106055593)

[Anexo: 6](#_Toc106055594)

[Kernel.c 6](#_Toc106055595)

[Función dormir: 6](#_Toc106055596)

[Cambio en int\_reloj descrito anteriormente: 6](#_Toc106055597)

[Funciones de mutex: 7](#_Toc106055598)

[Cambio en crear\_tarea: 11](#_Toc106055599)

[Funciones auxiliares: 12](#_Toc106055600)

[Kernel.h: 13](#_Toc106055601)

[Servicios.h: 14](#_Toc106055602)

[Serv.c: 14](#_Toc106055603)

[Llamsis.h: 14](#_Toc106055604)

# Llamada que bloquea al proceso un plazo de tiempo:

Lo primero que hacemos es añadir la rutina en kernel.c además de añadirlo en la tabla de servicios de kernel.h y sumar uno en llamsis.h en NSERVICIOS dejándolo a 5 luego añadir la interfaz en serv.c y añadir el prototipo de la llamada dormir a servicios.h.

A parte se ha añadido lista\_BCPs lista\_dormidos = {NULL,NULL}; en kernel.h para cómo se pide en el enunciado definir una lista de procesos esperando plazos y en la función int\_reloj se ha añadido el código que se ve en el anexo para contabilizar el tiempo que lleva dormido y cuanto le queda además de que lo despierta cuando llega este a 0.

Ahora se procederá a explicar brevemente el código del anexo:

Lo primero que hacemos es crear unas variables para guardar los segundos que se va a dormir el proceso y otra variable para el nivel de interrupción luego actualizamos el BCP al estado de bloqueo y con el numero de segundos que va a estar dormido, se guarda el nivel de interrupción y se fija en 3, tenemos que sacar de la lista los procesos listos que es el primer proceso y lo ponemos en la lista de bloqueados aquí he decidido restaurar el nivel de la interrupción anterior aunque creo que se puede realizar antes del return 0; y para terminar se obtienen un nuevo proceso actual con el planificador para realizar un cambio de contexto guardando el contexto del proceso que se va a dormir para restaurarlo mas tarde cuando se despierte.

Cuando se produce la interrupción del reloj se mira si el proceso es NULL si no se decrementa el tiempo en uno y se muestra cuanto tiempo le queda y se pasa al siguiente proceso cuando los procesos quedan a 0 se elevan las interrupciones en el reloj para pararlo y se dispone a despertar el proceso.

# Mutex:

Lo primero que hacemos es añadir las rutinas en kernel.c ademas de añadirlo en la tabla de servicios de kernel.h y sumar uno en llamsis.h en NSERVICIOS dejándolo a 10 luego añadir la interfaz en serv.c y añadir el prototipo de cada llamada a servicios.h.

Ahora se procederá a explicar brevemente el código del anexo:

## Función Crear\_mutex:

Miramos que no haya demasiados caracteres, que no hay otro con el mismo nombre y si se han alcanzado el número máximo de mutex creados en el sistema para esto se ha tenido que añadir el código en el anexo (Cambio en crear tarea en el índice) ya que si no se inicializaban los descriptores a -1 daba error en la prueba mutex 1 y 2 y decía que no había descriptores disponibles y en el caso de que dormir se bloquea el proceso hasta que se encuentre disponible un mutex después de esto se crea al mutex.

## Función abrir\_mutex:

Se abre el mutex ya creado, se busca si hay un descriptor libre y una vez encontrado el descriptor libre se guarda, se asigna y se actualizan las variables.

## Función lock:

Se lee el descriptor del mutex, se mira si el mutex ha sido abierto anteriormente y si no da error, ahora miramos si el mutex es recursivo si lo es comprobamos si ha sido bloqueado por el proceso actual si ha sido bloqueado el proceso actual es el dueño y se puede volver a bloquear si no ha sido bloqueado se tiene que bloquear el proceso porque estaríamos en la cola del mutex aplicando algo parecido a lo que hacemos en dormir, si resulta que el mutex no es recursivo se mira si ha sido bloqueado por el proceso actual si esta bloqueado se sale si no el mutex ha sido bloqueado por otro proceso con lo que hay que bloquear el proceso actual utilizando algo parecido a dormir y por ultimo si el mutex no está bloqueado se bloquea y si ha salido todo bien y sin errores se asigna el proceso de ese momento como propietario del mutex.

## Función unlock:

Desbloquea el mutex bloqueado por lock que al igual que en lock solo puede desbloquearlo el dueño del mutex.

El código de lock es igual que lock lo único que en vez de bloquear el mutex se desbloquea, pero esencialmente es lo mismo como podemos comprobar en el anexo con el código.

## Función Cerrar\_mutex:

El mutex se cierra cuando el proceso no lo necesita si el mutex estuviera bloqueado este se desbloquea y en caso de ser recursivo se desbloqueará las veces que haga falta y en caso de que no haya procesos que hayan abierto mutex este será borrado.

## Funciones auxiliares:

descriptor\_libre(): Busca y devuelve la posición libre en el array de descriptores de cada proceso del sistema.

descriptor\_mutex():Busca y devuelve los mutex libres.

buscar\_nombres\_iguales(char\* nombre): Busca en todos los mutex en busca de un nombre igual para generar un error

# Round Robin:

No se ha podido implementar este apartado.

# Ejemplos de ejecución:

## DORMIR:

Texto

Descripción generada automáticamente

Imagen de la pantalla de un computador

Descripción generada automáticamente con confianza baja

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

## Prueba mutex 1 y prueba mutex 2 resultados:

Texto

Descripción generada automáticamente

(no se porque aparece una excepción de memoria pero por lo demás funciona todo bien)

# Anexo:

Se aporta capturas de imagen de los cambios realizados en cada archivo.

## Kernel.c

## Función dormir:

Texto

Descripción generada automáticamente

## Cambio en int\_reloj descrito anteriormente:

Texto

Descripción generada automáticamente

## Funciones de mutex:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

## Cambio en crear\_tarea:

Texto

Descripción generada automáticamente

## Funciones auxiliares:

Texto

Descripción generada automáticamente

## Kernel.h:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

## Servicios.h:

Texto

Descripción generada automáticamente

## Serv.c:

Texto

Descripción generada automáticamente

## Llamsis.h:

Texto

Descripción generada automáticamente