SISTEMAS OPERATIVOS AVANZADOS
AÑO 2007
TRABAJO PRACTICO N° 3

Titulo: Exclusión Mutua Distribuida - Algoritmo de Raymond

Formato de entrega

Deberá entregar:

1. Documento IMPRESO y en carpeta en formato A4 conteniendo las siguientes secciones:
   - 1.1. Carátula: Debe contener como MINIMO:
     - 1.1.1. Materia
     - 1.1.2. Numero y Título de trabajo práctico
     - 1.1.3. Apellido y Nombres de los integrantes
     - 1.1.4. Direcciones de correo electrónico (de todos los que tengan una cuenta habilitada)
     - 1.1.5. Año de cursado
   - 1.2. Cuerpo: Con la resolución de los Items del TP. Cuando el TP implique la modificación o el desarrollo de nuevo código, debe incluirse dicho código. Asumir que el documento se utilizará para un eventual debug. Incluir todas las aclaraciones que se crean necesarias para ayudar en la corrección
   - 1.3. Descargos: Indicar aquí las razones por las que no se pudo concluir o realizar alguno de los Items del TP. Si existiesen dudas acerca de la forma en que se resolvió algún ítem, mencionarlo también en esta sección.
2. Diskette:
   - 2.1. Formato: ext2
   - 2.2. Contenido:
     - 2.2.1. src : Conteniendo todo el código fuente desarrollado y/o modificado.
     - 2.2.2. obj : Conteniendo todos los ejecutables.
     - 2.2.3. doc: Conteniendo toda la documentación. Incluya aquí una copia del documento del que se habla en el punto 1.
     - 2.2.4. src/Makefile: En este archivo de reglas para *make*, incluya los comandos necesarios para que quien corrija el documento pueda correr los comandos:
       - 2.2.4.1. *make*: Esto debe compilar todos los programas necesarios y dejar los objetos en el directorio *obj*.
       - 2.2.4.2. *make clean*: Esto debe eliminar todos los programas objetos correspondientes del directorio *obj*
       - 2.2.4.3. make copy: Esto debe copiar TODO el diskette al directorio /home/soa/2007/tp3/NOMBREGRUPO, donde NOMBREGRUPO es el nombre asignado al grupo. El árbol de directorios debe ser creado si no existiese.

Objetivo:

El objetivo es implementar exclusión mutua distribuida y lograr experiencia técnica en el diseño de sistemas distribuidos.

Descripción:

Deberá implementar el algoritmo de Raymond utilizando sockets.

Proceso para verificación del funcionamiento:

Para comprobar el funcionamiento del algoritmo, se creará un programa con el siguiente formato:

 # *mutexloop <port_server>*

siendo **port_server**: el puerto del servidor ***mutexd*** correspondiente

Todo proceso ***mutexloop*** consiste básicamente en:

```
while(TRUE)
        {
        printf("Estoy en la FUERA de la Región Crítica\n");
        printf("Presionar una tecla para Ingresar a la RC...\n”);
        getc(); /* leer del teclado */
        Entrar_RC();
        printf("Estoy en la DENTRO de la Región Crítica\n");
        printf("Presionar una tecla para Salir de la RC...\n”);
        getc(); /* leer del teclado */
        Salir_RC();
        }
```

Las funciones ***Entrar_RC()*** y ***Salir_RC()*** deberán desarrollarse solicitando el ingreso y la salida de RC a un servidor de exclusión mutua **local** denominado **mutexd** utilizando sockets UDP.

ATENCION: debe haber un proceso **mutexd** por cada proceso **mutexloop**.

A los efectos de simplificar la implementación, todos los procesos se ejecutarán en la misma Computadora. Esto trae como ventaja adicional de que se utiliza la misma base de tiempo para todos los procesos necesarios para los timestamps requeridos en la implementación del algoritmo.

En el archivo ***/etc/mutex.cfg*** se configurarán los puertos de los distintos servidores ***mutexd***:

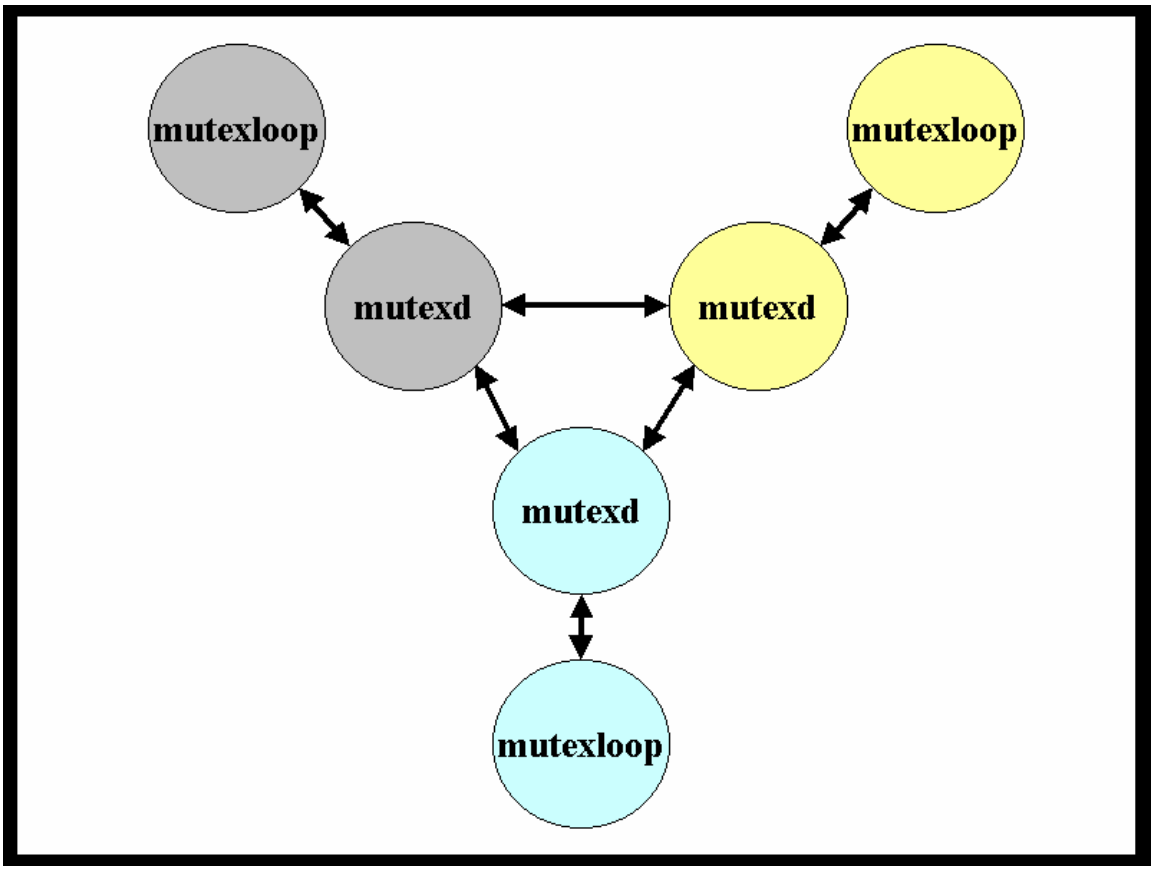
Ej:

5001 mutexd1

5002 mutexd2

5003 .....

5007 mutexd7



Formato del servidor de exclusión mutua:

 # *mutexloop <port>*

siendo **port**: el puerto donde el servidor escuchará las peticiones de su ***mutexloop*** y de otros ***mutexd***.

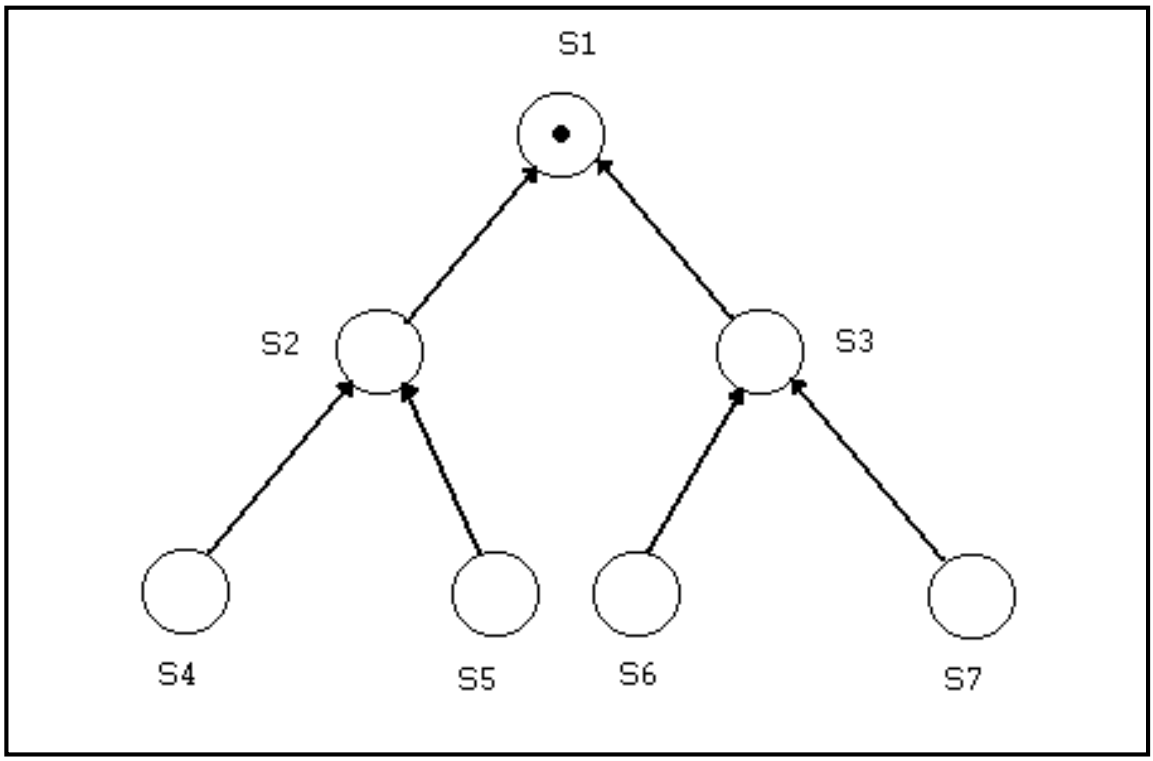
Se sugiere desarrollar a ***mutexd*** como una máquina de estados que puede recibir:

- paquetes desde la red provenientes desde los otros ***mutexd***
- paquetes o peticiones del **mutexloop** propio

Algoritmo para mutexd

INICIALIZACION

- Enviar un mensaje de tipo HELLO a todos los servidores.
- Cuando un servidor **i** recibe un mensaje HELLO de un servidor **j**, marca al servidor **j** como ONLINE y responderle con otro HELLO.
- Recién cuando el servidor **i** tiene marcado en estado ONLINE a todos los servidores, puede comenzar el algoritmo.
- Se constituirá una estructura de arbol, como lo muestra la figura. En un principio el servidor 1 (uno) será el poseedor del token.



EJECUCION

Si se recibe una petición del **mutexloop** de tipo **ENTER_RC**

- Enviar un mensaje REQUEST a todos los otros procesos **mutexd en camino hacia el token.** Agrega su petición en su **request_q.**
- Cuando sitio en el camino hacia el TOKEN recibe REQUEST, lo inserta en su **request_q** y envía un REQUEST al nodo en el camino hacia el TOKEN.
- Cuando el **root** recibe el REQUEST le envía a través de los nodos intermedios el TOKEN y fija su variable **holder** apuntando a ese sitio.
- Cuando un sitio recibe el TOKEN:
- suprime la entrada al tope de su **request_q**.
- envía el TOKEN hacia el sitio destino.
- fija su variable **holder** hacia ese sitio.

- Si la **request_q** no esta vacía, envía un REQUEST al sitio apuntado por holder.
- Si entra a la RC si dispone del TOKEN y su petición está en la primer entrada de la **request_q**. Si es así, puede entrar a la región critica y suprimir su entrada de la cola
- Si se recibe una petición del proceso local **mutexloop** de tipo **LEAVE_RC**
- Si la **request_q** no está vacía:
- Suprime el primer elemento
- Envía el TOKEN a ese sitio
- Apunta **holder** a ese sitio
- Si luego, la request_q no está vacía:
- Envía un REQUEST al sitio apuntado por holder

Es importante, a los efectos didácticos y de depuración, que se presenten en la pantalla de cada servidor **mutexd** los mensajes enviados y recibidos como así tambien los estados en que se encuentran la máquina.

Sistemas Operativos: Linux
Lenguaje: C
Bibliografia:

- CD-ROM de la Cátedra
- http://www.csulb.edu/~ykotow/raymond.htm
- http://nike.cecs.csulb.edu/~ktarazi/index.html
- http://courseweb.sp.cs.cmu.edu/~cs612/projects/proj2/proj2.pdf
- http://www.cs.colostate.edu/~cs551/CourseNotes/Synchronization/RaymondTreeExample.html
- http://www.cse.msu.edu/~sandeep/cse812fall2002/Slides/synchronization.pdf