UNIVERSIDAD CARLOS III DE MADRID
AREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES
GRADO EN INGENIERÍA INFORMÁTICA. SISTEMAS DISTRIBUIDOS

Para la realización del presente examen se dispondrá de **1 hora y 15 minutos. NO** se podrán utilizar libros, apuntes **ni** calculadoras de ningún tipo.
Alumno: _____**Grupo:** _____

Ejercicio 1 (2,5 puntos). Responda a las siguientes preguntas cortas justificando brevemente su respuesta:

a) Describir brevemente qué es un servicio web.
Un servicio web es una colección de protocolos y estándares abiertos que sirven para intercambiar datos entre aplicaciones:
- Escritas en distintos lenguajes de programación
- Ejecutan en distintos sistemas operativos y arquitecturas
- Desarrolladas de manera independiente

Los servicios web permiten adaptar la programación web a aplicaciones no basadas en navegador.

b) En una transacción HTTP identificar los principales campos de la línea de petición y de respuesta.
HTTP (HyperText Transfer Protocol) es el protocolo de Internet que permite la transferencia de hipertexto (páginas que contienen hiperenlaces). Utiliza el modelo cliente-servidor, y se basa en una petición y una respuesta:
- Mensaje de petición:
  Contiene una línea de petición, un conjunto de cabeceras y una línea en blanco:

  <Método><espacio><URIsolicitado><espacio><protocolo>
  <Cabeceras>
  \r\n

  El método identifica la operación (GET, PUT, etc.), el URI identifica unívocamente el recurso en Internet al que se desea acceder y el protocolo identifica la versión de HTTP (1.0, 1.1, ó 1.2).
  Las cabeceras son opcionales para el mensaje de petición (Host, User-Agent, etc.)

- Mensaje de respuesta:
  Contiene los siguientes campos:
  <Protocolo><código>
  <Cabeceras>
  <recurso>

El campo **protocolo** identifica la versión de HTTP utilizada por el servidor. El campo **código** es un identificador de 3 caracteres numéricos que sirve para representar el éxito o fallo en la transacción. En la siguiente línea aparecen las cabeceras necesarias para el envío del recurso (Lenght,Content-Type, etc.) y finalmente se envía el recurso solicitado (en su caso).

c) Definir brevemente en qué consiste el multicast atómico.

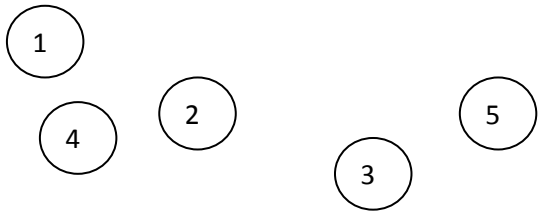
El multicast atómico es un tipo de multicast que se caracteriza por dos características: fiabilidad y orden de los mensajes. La primera de ellas garantiza que todos los mensajes lleguen a todos los integrantes del grupo multicast (o no lleguen a ninguno); la segunda garantiza que la entrega de los mensajes se realiza en el mismo orden para todos los integrantes del grupo multicast.

d) ¿Cuál es la función del servicio *portmapper*?

El servicio portmapper es el servicio de registro de las RPCs de SUN. Este servicio ejecuta en un puerto fijo, el 111, y tiene dos funciones principales:

- Registrar una RPC. Los servidores de RPC se conectan al servicio portmapper para obtener un puerto libre en el que escuchar las peticiones de los clientes de RPC. De esta manera dan de alta los servicios remotos.
- Localizar el puerto donde ejecuta un servidor RPC. Los clientes de RPC se conectan al servicio portmapper para conocer el puerto dónde ejecuta el servidor de RPC, para poder posteriormente solicitar un procedimiento remoto.

e) Dado el siguiente conjunto de n nodos, con n=5, que usa el método de votación dinámica (quórum) para mantener la consistencia de las réplicas.



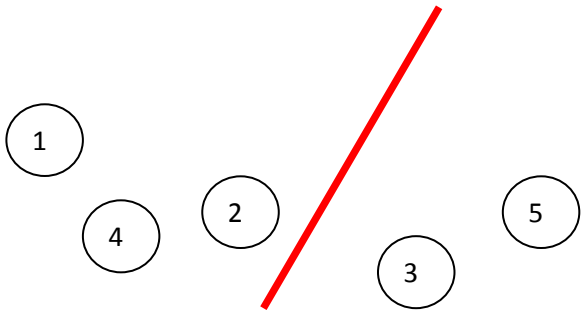
Suponga que en un momento de la ejecución se produce un fallo en la red que da lugar a dos particiones: {1,2,4} y {3,5}. En ese momento los nodos tienen los siguientes valores NV y SC para una determinada réplica:

	Nodo 1	Nodo 2	Nodo 3	Nodo 4	Nodo 5
SC	5	5	5	5	5
NV	3	3	3	3	3

NV= número de versión SC=cardinalidad de actualización

En esta situación: ¿se podría actualizar en la partición {1,2,4}? Justifique su respuesta.

Solución:



En la partición {1,2,4} se quiere actualizar una réplica y se ejecuta el algoritmo de votación dinámica.

N=max{NVi}=max{3,3,3}=3
I={1,2,4}
M=max{SCi}=max{5,5,5}=5
If (|I|>N/2) → se puede actualizar
Como |I|=3 y es mayor que 5/2 sí se puede actualizar en la partición {1,2,4}.

	Nodo 1	Nodo 2	Nodo 3	Nodo 4	Nodo 5
SC	3	3	5	3	5
NV	4	4	3	4	3

f) Un cliente de NFS realiza la siguiente operación de apertura de un fichero remoto:
   - fd=open("/home/user/data.txt");

Describir el conjunto de llamadas NFS necesarias para abrir ese fichero en el servidor NFS.

Los servicios que proporciona el servidor NFS se ofrecen en forma de RPC. Los clientes NFS deben traducir los servicios sobre ficheros a las llamadas RPC equivalentes que son enviadas por la red al servidor NFS. En particular, el servicio open se mapea a varias llamadas al procedimiento remoto LOOKUP, una por cada directorio/fichero de búsqueda. Este procedimiento devuelve el filehandle (manejador de fichero remoto) y atributos asociados al campo de búsqueda.

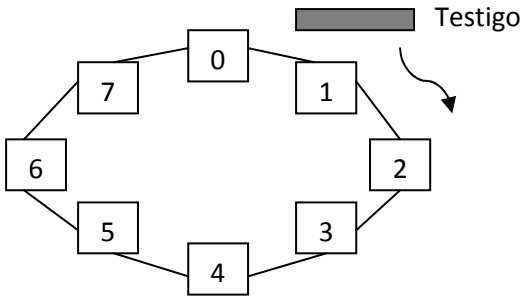
lookup(rootfh, "/home") devuelve **(fh0, attr0)**
lookup(fh0, "/user") devuelve **(fh1,attr1)**
lookup(fh1, "/data.txt") devuelve **(fh2,attr2)**

El filehandle fh2 correspondiente a data.txt es el manejador usado en siguientes operaciones sobre ese fichero.

g) ¿Cuál es la función del servicio *mount*?

El servicio mount se utiliza para montar un árbol de ficheros exportado por el servidor NFS en una ubicación (punto de montaje) del cliente de NFS. Montar en este contexto significa proyectar la imagen del árbol de directorios del servidor NFS en el cliente NFS, es decir visualizar el árbol de directorios del servidor en la máquina local (no es una copia).

Ejercicio 2 (3 puntos). Se desea usar el algoritmo del anillo con testigo para implementar exclusión mutua en un sistema local. En el algoritmo del anillo con testigo un conjunto de N procesos (i = 0.. N-1) se organizan en forma de anillo, de manera que un proceso i sólo puede comunicar con el proceso i+1 y el proceso i-1. Por el anillo circula un testigo (token) de manera que sólo puede entrar a ejecutar la sección crítica (SC) aquel proceso que tenga el testigo. El sentido en que circula el anillo es unidireccional. Si un proceso recibe el testigo pero no quiere entrar a ejecutar en la sección crítica, reenvía el testigo al siguiente proceso. Si por el contrario quiere entrar en la SC debe esperar a recibir el token; una vez obtenido entra en la SC y a la salida simplemente reenvía el token al siguiente proceso. Se pide implementar exclusión mutua usando el algoritmo previamente descrito y las colas de mensajes de UNIX. Considere que comienza a ejecutar el proceso con identificador 0.



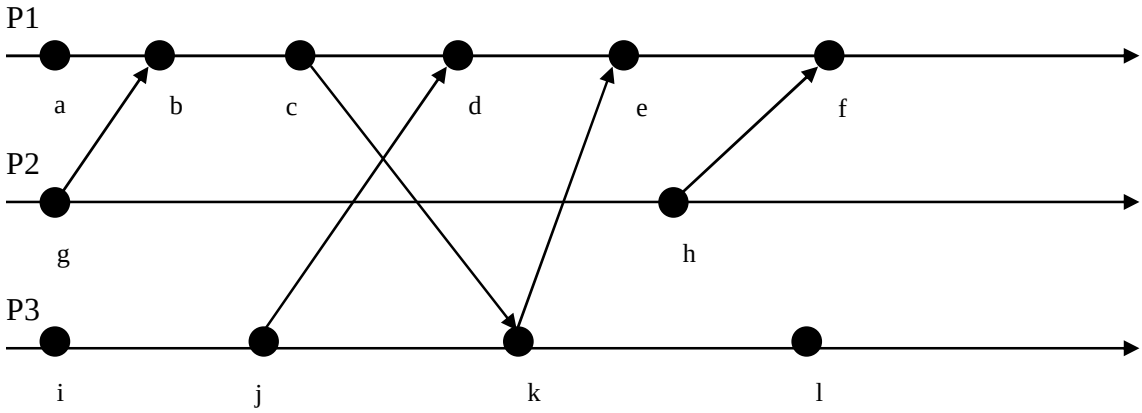
Una posible solución es la siguiente. Cada uno de los procesos va a tener su propia cola de mensajes. Cada proceso puede acceder a su cola de mensajes para extraer el token y la cola de mensajes del proceso siguiente en el anillo para enviarle el testigo. Uno de los decisiones de diseño es el nombrado de las colas para que los procesos puedan acceder a las colas de los procesos siguientes en el anillo. Nombraremos las colas como “queue-X” donde X es el identificador del proceso.

```
#define    N    8
pthread_mutex_t mutex;
pthread_cond_t copiado;
int seguir = FALSE;

void main(){
     pthread_t threads[N];

     pthread_mutex_init(&mutex);
     pthread_cond_init(&mutex);

     // Crear los procesos: los consideramos dependientes
     for (int i=0;i<N;i++){
```

```
            pthread_mutex_lock(&mutex);
            pthread_create(&threads[i], NULL, anillo, &i)
            while(!seguir){
                  pthread_cond_wait(&mutex,&copiado);
            }
            seguir = FALSE;
            pthread_mutex_unlock(&mutex);
      }
      for (i=0;i<N;i++){
            pthread_join(threads[i]);
      }
      pthread_mutex_destroy(&mutex);
      pthread_cond_destroy(&copiado);
}


void anillo(int *id){
      int mi_id;
      int token = 1;
      mqd_t queue_token;
      mqd_t queue_vecino;
      struct mq_attr atributos;
      char mi_nombre[10],nombre_vecino[10];

      pthread_mutex_lock(&mutex);
      mi_id = *id;
      seguir = TRUE;
      pthread_mutex_unlock(&mutex);

      // Creamos nuestra cola de mensajes
      atributos.mq_maxmsg = 1;
      atributos.mq_msgsize = sizeof(int);

      sscanf("%s-%d", mi_nombre, "queue", mi_id);
      queue_token=mq_open(mi_nombre,O_CREAT|O_RDONLY,0777,
                                          &atributos);

      // Abrimos la cola de mensajes de lectura del siguiente
      proceso en el anillo
      sscanf("%s-%d", nombre_vecino, "queue", mi_id+1);
      queue_vecino = mq_open(nombre_vecino, O_WRONLY, 0777,
                                          &atributos);

      if (mi_id  == 0){
            // Insertamos el token si somos el primer nodo
            mq_send(queue_token,(char*)&token, sizeof(int),0);
      }

      while(1){
            // Vemos si tenemos el token en nuestra cola
            mq_receive(queue_token,(char*)&token, sizeof(int),0);

            // Si tengo el token, entro en la sección crítica y
            lo reenvío al siguiente proceso
            <Ejecutar SC>
            mq_send(queue_vecino,(char*)&token, sizeof(int) ,0);
```

```
        }
        mq_close(queue_token);
        mq_close(queue_vecino);
        mq_unlink(queue_token);
        pthread_exit();
}
```

Ejercicio 4 (1.5 puntos). El siguiente diagrama muestra una serie de eventos que ocurren entre estos tres procesos. Estos eventos tienen una marca de tiempo asignada (*timestamp*).



Se pide:

a) Utilizando relojes lógicos de Lamport, indique las marcas de tiempo para los eventos que aparecen en la figura.
b) ¿Qué significa que dos eventos sean concurrentes? Indicar cuáles son los eventos concurrentes en el ejemplo de la figura.
c) Responda a las siguientes preguntas:
    1. ¿Obtenemos una relación de orden total con los relojes de Lamport? Razone la respuesta.
    2. Si no es así, ¿cómo podemos conseguir que exista una relación de orden total entre los procesos?
d) Utilizando relojes vectoriales, indique las marcas de tiempo para los eventos que aparecen en la figura.

Solución:

a) P1 a=1,b=2,c=3,d=4,e=5,f=6
   P2 g=1,h=2
   P3 i=1,j=2,k=4,l=5

b) Se dice que dos eventos a,b son concurrentes cuando no se puede demostrar la relación de causalidad potencial “precede a” entre ellos. Es decir no podemos demostrar que a→b ni b→a. Cuando ocurre esto se denota mediante a||b.

a||g, a||i, a||j, b||i, b||j, b||h, c||i, c||j, c||h, d||h, e||h, g||i, g||j, a||h, b||h, c||h, d||h, e||h, h||i, h||j, h||k, h||l, f||l, d||l, e||l, f||l

c) Con los relojes de Lamport podemos obtener un orden parcial. Con este orden parcial algunos eventos del sistema podrían aparecer no ordenados y por tanto ser concurrentes con otros. Para poder obtener una relación de orden total entre los procesos se pueden añadir a las marcas de tiempo el identificador del proceso al que pertenece el evento.
d) P1 a=(1,0,0), b=(1,2,0), c=(2,2,0), d=(2,2,3), e=(2,2,4), f=(2,3,4)
P2 g=(0,1,0), h=(0,2,0)
P3 i=(0,0,1), j=(0,0,2), k=(2,2,3), l=(2,2,4)