Taller de Sistemas Operativos Avanzados
Práctica #1: Modelo Cliente / Servidor
"Paso de Mensajes entre Procesos"

Objetivo

Implementar una simulación de los programas cliente y servidor que serían ejecutados sobre un sistema operativo basado en micro núcleo, el cual proporcionara los servicios de comunicación básica mediante operaciones *send*() y *receive*() para paso de mensajes entre procesos.

Descripción

En un sistema, en cuanto al concepto *servicio* se refiere, existen dos tipos de procesos involucrados: los clientes y los servidores: el *cliente* solicita un servicio al *servidor* mediante la construcción y envío de un mensaje *solicitud* con cierta estructura y tipo de contenido, lo cual ya está definido para el *servicio* desde el diseño del *servidor*, cuando éste recibe la solicitud, la interpreta, realiza el *servicio* y construye un mensaje *respuesta*, el cual envía al *cliente*, finalmente este último al recibir la *respuesta*, extrae los datos solicitados y continúa su trabajo; el paso de mensajes entre clientes y servidores es sustentado por el micro núcleo, mediante sus operaciones send(dest,messagePtr) y receive(addr,messagePtr).

Requerimientos Generales

La simulación del modelo Cliente-Servidor deberá cumplir con lo siguiente:

1. Aplicación para ambiente gráfico desarrollada en un lenguaje de programación que soporte manejo de hilos (threads) así como comunicación por red no orientada a conexión (datagramas)
2. Adecuar el diseño arquitectónico e implementación de la aplicación con el patrón Model-View-Controller, para esto los archivos que contengan el código fuente correspondiente al *núcleo de la aplicación* (requerimientos no funcionales) debe no hacer referencia a tipos de datos relacionados directamente con los requerimientos de la *vista*.

Requerimientos de la Vista y Funcionales

3. Creación de ventanas independientes para los tres tipos de entidades en el sistema (núcleo, proceso cliente y proceso servidor) y que cuenten cada una con un área de texto donde se permita imprimir cada *suceso significativo* en que se encuentre involucrada cada entidad respectivamente. Para esta práctica, las ventanas de los procesos deberán mostrar:

a) Ambos procesos: "Inicio de proceso"
b) Ambos procesos: "Invocando a receive()"
c) Ambos procesos: "Generando mensaje a ser enviado, llenando los campos necesarios"
d) Ambos procesos: "Señalamiento al núcleo para envío de mensaje"
e) Proceso servidor: "Procesando petición recibida del cliente"
f) Proceso servidor: El nombre de la operación solicitada por el cliente y los datos proporcionados para la operación a realizar
g) Proceso cliente: "Procesando respuesta recibida del servidor"
h) Proceso cliente: El resultado de la operación
i) Núcleo: notificación del inicio y fin de cada proceso

4. La aplicación inicie mostrando la interfaz gráfica (IG) correspondiente al núcleo.
5. A partir de la IG del núcleo se permita iniciar las IGs respectivas de los procesos clientes y servidores (al menos para 100 procesos).
6. Sea visible el ID de proceso tanto del proceso cliente como del servidor en sus respectivas IGs.
7. La aplicación permita cerrar las IGs correspondientes a los procesos (cliente o servidor) de forma que se puedan iniciar de nuevo sin terminar la aplicación (sin cerrar la IG del núcleo)
8. La IG de núcleo permita introducir una dirección IP y un ID de proceso a usar por los requerimientos no funcionales de la práctica #2.

Requerimientos No Funcionales

9. La ejecución de los procesos simulados deberá ser concurrente, esto es, que la serie de instrucciones de cada uno cuente con su propio hilo de control.
10. El núcleo asigne un identificador (ID) de proceso único (un entero de 4 bytes) y secuencial a cada proceso que sea iniciado.
11. El núcleo tenga la facultad de iniciar y terminar procesos indefinidamente (al menos 100 procesos), dándolos de alta o baja de sus registros.
12. Los mensajes que los procesos han de enviarse entre sí, sean en forma de un arreglo de bytes, el cual tenga la estructura siguiente:

A) Para un proceso cliente emisor:
a) Primeros 4 bytes: Identificador del emisor (ID del cliente, reservado para el núcleo)
b) Siguientes 4 bytes: Identificador del receptor (ID del servidor; reservado para el núcleo)
c) Siguientes 2 bytes: Código de operación (CODOP)
d) Siguientes 1014 bytes: Datos relativos a la operación solicitada; puedan ser desde 1 hasta 1014, dependiendo de la información a enviar y reservando de la memoria únicamente los bytes necesarios para los datos a enviar.

B) Para un proceso servidor emisor:

a) Primeros 4 bytes: Identificador del emisor (ID del servidor, reservado para el núcleo)
b) Siguientes 4 bytes: Identificador del receptor (ID del cliente, reservado para el núcleo)
c) Siguientes 1016 bytes: Datos relativos a la respuesta a enviar; al igual que para el cliente, reservando de la memoria únicamente los bytes necesarios.

+ Los primeros 8 bytes del mensaje a enviar serán tratados en la práctica siguiente

13. El núcleo incluya las primitivas send(dest,messagePtr) y receive(addr,messagePtr), a través de las cuales se transfieran los mensajes entre cliente y servidor

a) *dest* sea el identificador del proceso destinatario (un entero de 4 bytes) más no su ID de proceso (este parámetro será usado a partir de la práctica siguiente)
b) *addr* sea el ID del proceso que espere recibir un mensaje (este parámetro también será usado a partir de la práctica siguiente)
c) *messagePtr* sea el apuntador a un arreglo de bytes definido en el espacio del proceso (cliente ó servidor).

14. El proceso cliente solicite un servicio (simbólico al menos) al proceso servidor, enviándole un mensaje con un CODOP y datos relativos a este asignables desde la IG (del cliente); una vez enviada la solicitud, el cliente espere la recepción de la respuesta a enviar por el servidor; una vez recibida la respuesta, procese el contenido y muestre el resultado en la interfaz gráfica.
15. El proceso servidor, después de haberse iniciado, espere solicitudes de clientes y que tenga capacidad de atender peticiones distintas (no es necesario que sea simultáneo) en base al campo CODOP y datos de la solicitud recibida, y que elabore y envíe al cliente un mensaje de respuesta acorde a la operación solicitada y datos proporcionados, de tal forma que el código fuente encargado de la interpretación y procesamiento de la solicitud, así como la elaboración de la respuesta, sea independiente para cada CODOP.
16. El servidor pueda atender varias solicitudes, desde la primera vez y después de enviada la respuesta a la solicitud anterior y sin tener que terminar la aplicación (sin cerrar la IG del núcleo) ni la ventana del servidor para ello.
17. Las actividades del cliente (elaboración de solicitud, envío de solicitud, recepción de respuesta, procesamiento de respuesta) y las del servidor (recepción de solicitud, procesamiento de solicitud, elaboración de respuesta, envío de respuesta) sean llevadas a cabo por el hilo de control correspondiente al proceso (no por un hilo generado como evento desde la IG).
18. Los mensajes entre procesos sean enviados y recibidos solamente haciendo uso de las primitivas send y receive proporcionadas por el núcleo.
19. El kernel y sus componentes estén protegidos de acceso público (encapsulamiento) ante los procesos de usuario ordinarios, para los cuales no haya sido diseñado explícitamente un procedimiento o función dedicados al servicio de procesos en general (como send y receive)
20. Los procesos y el kernel no compartan el uso de variables que no sean por paso de parámetros
21. Las primitivas send y receive sean declaradas como procedimientos, no como funciones
22. El proceso que invoque a send recupere el control después de que haya sido enviado el mensaje al otro proceso de modo que se cumpla con primitivas por bloqueo

23. El proceso que invoque a receive, recupere el control después de que el mensaje recibido haya sido copiado a su espacio de memoria (primitivas por bloqueo)
24. Las primitivas send y receive hagan uso de los parámetros recibidos (sólo de messagePtr en está práctica) y únicamente utilicen variables externas al procedimiento si tienen que ver con alguna estructura manejada por el núcleo o alguna variable de estado propiedad del mismo
25. La comunicación entre los procesos requiera del paso de mensajes a través del núcleo; la comunicación por red no es requerimiento de esta práctica.
26. La comunicación entre procesos es suficiente para esta práctica con sólo la ejecución simultánea de un proceso cliente y un proceso servidor, para un instante dado, y con la posibilidad de sustituir un servidor por otro, previo terminar al anterior, o un cliente por otro de la misma forma.

Requerimientos y Restricciones Complementarios

27. No se use el constructor String(byte[]) a menos que todos los elementos del arreglo sean valores ASCII interpretables como caracteres visibles, en su lugar se puede utilizar el String(byte[],int,int)
28. En cuanto a los *sucesos significativos*, es imprescindible que las instrucciones para imprimir sucesos distintos no sean consecutivas.
29. En cuanto a los *sucesos significativos*, para aquellos que hacen referencia al presente y presente continuo (gerundios), se impriman una línea antes de donde ocurre la primera instrucción relacionada; para un suceso ocurrido el pasado, la instrucción de impresión sea en línea inmediata siguiente a donde se procesó lo ocurrido.
30. Borrar cualquier dato almacenado en tablas cuyo uso posterior no esté definido
31. Todo parámetro en una función, procedimiento o método debe ser utilizado (a menos que alguno de los requerimientos de la práctica indique lo contrario)
32. El funcionamiento de los procesos debe ser independiente del núcleo en cuanto a que éste último requiera invocar métodos albergados en los procesos
33. Todo valor asignado a una variable debe ser necesario para la operación de la alguna parte de la práctica
34. El código fuente sea libre de errores de compilación y advertencias

Criterios de Evaluación

? Los establecidos en las “Reglas de Operación y Evaluación” del Taller de Sistemas Operativos Avanzados y los correspondientes “Periodos de Entrega” de la práctica.
? Código fuente indicando en la parte superior de cada archivo: nombre del(la) alumno(a), sección y no. de práctica; entregado vía e-mail en formato zip.
? Fecha de asignación de la práctica: _09_ de _Marzo_ de 2009
? Fecha límite para entrega de la práctica: _18_ de _Marzo_ de 2009
? Observación: _____
? Calificación en base a cobertura de requerimientos y fecha de entrega