Universidad Tecnológica Nacional – Facultad Regional Buenos Aires

Ingeniería en Sistemas de Información

sistemas operativos

INSOMNIO

Restricciones tecnológicas

Documentación

Idea y diseño:
Ing. Diego Marafetti

Colaboradores:
Pablo Lorenzatto
Santiago Ciciliani
Ing. Pedro Vazquez

Revisión: 1.0
2° Cuatrimestre 2010

Requerimientos técnicos y limitaciones

Introducción

Este documento contiene una descripción de cómo utilizar todas las tecnologías que son requeridas para llevar a cabo la implementación del sistema distribuido. Contiene una serie de restricciones que el alumno deberá aplicar en sus soluciones.

Programación en C++

Para el desarrollo en la plataforma Windows solo estará permitido el uso de la Standard C++ Library (Ver Anexo Documentación). Toda biblioteca externa, no especificada en este documento queda prohibida. Esto garantiza que los desarrollos sean reproducibles en todos los ambientes de trabajo de la cátedra. Un ejemplo de bibliotecas no permitidas son: MFC (Microsoft Foundation Class Library) y .NET Framework.

Dynamic Link Libraries - Windows

La creación de la Dll se realizará desde un proyecto nuevo en Visual Studio. El tipo de proyecto será para la creación biblioteca de enlace dinámico y estático. Como opción adicional no se exportarán símbolos.

Propiedades del proyecto

El proyecto del VS deberá ser configurado con las siguientes propiedades:

Propiedades C/C++:

- Runtime Library: Multi-Thread (/MT)
- Calling Convention: __cdecl (/Gd)
- Compile As: Compile as C Code (/TC) ó Compile as C++ Code (/TP) según corresponda.

Exportación de funciones

La forma de exportación se hará utilizando el modificador *__declspec(export)* por lo que no será necesario utilizar un module definition file (.def). La Dll posee una tabla de exportación con los nombres de todas las funciones que la misma exporta a otros ejecutables. Esta tabla se podrá consultar desde la herramienta DUMPBIN con la opción /EXPORT que nos brinda el Visual Studio.

Linking

El método de enlace que se usará en la aplicación será el enlace explicito. Por esto no será necesario enlazar la aplicación con ninguna import library (.lib). El enlace será hecho por la aplicación en run-time y para lograrlo se utilizarán las llamadas: LoadLibrary, GetProcAddress, FreeLibrary.

Shared Objects – Entorno Unix

Las Shared Libraries serán cargadas cuando en programa se inicie y se deberán seguir las convenciones que aquí se mencionan. No se utilizarán bibliotecas de carga dinámica.

La biblioteca que se cree deberá tener el correspondiente "soname" con número de versión que corresponde con la versión de su interface. Deberá tener el "real name" con el número de realease actual. Adicionalmente deberá presentar el "linker name".

Comunicaciones – Sockets

Para la comunicación cliente/servidor se utilizarán los protocolos especificados en el trabajo práctico y se implementarán utilizando sockets (modelo Berkeley) orientados a la conexión del tipo *AF_INET* para IPv4 (en las tres plataformas).

Para la programación de sockets en Windows, se va a utilizar la biblioteca Winsock declarada en *winsock2.h*. Se requiere linkear con la dependencia *Ws2_32.lib*.

Inicialización de la biblioteca Winsock

Antes de llamar a cualquier función de la biblioteca Winsocks, se deberá inicializar dicha librería indicando la versión de Window Sockets y obteniendo los detalles de la implementación.

Finalizada la utilización de la biblioteca ó ante algún error en la inicialización de la misma, se deberá liberar los recursos y finalizar el uso de la biblioteca Winsock.

Esta inicialización se realiza **solamente una vez** por ejecución de la aplicación y no debe ser inicializada desde la Dll.

Los parámetros inicialización serán los siguientes:

Parámetros

Versión solicitada

Se utilizará la versión 2.2.

Process Management y Multithreading

Windows

Para la creación de threads en Windows se utilizará la función *_beginthreadex* perteneciente a la *C/C++ Runtime library*. (Para investigar: ¿cuáles son las causas de este comportamiento al usar la API *CreateThread*?).

Cada thread tendrá por defecto un *stack space* de 1024 kilobytes.

Solaris y Linux

En Linux, en caso de ser requerido, solo se podrá utilizar la API de la biblioteca pthreads que forma parte del estándar POSIX.

Para la plataforma Solaris se utilizará la API de Solaris Threads para la creación y manipulación de hilos de ejecución. Se puede dar el caso en que exista funcionalidad que no esté presente en Solaris Threads y si exista en POSIX. En este caso es posible utilizar ambas para adquirir funcionalidad extra. En caso de tener portabilidad de código en Linux y Solaris como requerimiento, estará permitido usar el estándar POSIX.

Cada Thread que la aplicación genere deberá tener por defecto el *stack space* en 1024 kilobytes por arriba del *PTHREAD_STACK_MIN*.

Para la creación de procesos, en caso de ser requerido, se utilizará el modelo *Fork-one Model*.

Thread Synchronization

En Windows se utilizarán Kernel Objects para la sincronización. Estos objetos serán los utilizados por las llamadas *wait functions* para coordinar la ejecución de múltiples threads al pasar de un estado señalizado a uno no señalizado.

A continuación se dará una descripción de los mecanismos que ofrece la Windows API y que estarán permitidos utilizar para la sincronización de threads en *user-mode*.

Los siguientes kernel objects podrán ser utilizados para la sincronización y queda a criterio del alumno cual utilizar según crea conveniente:

- *Events Kernel Objects*
- *Semaphore Kernel Objects*
- *Mutex*
- *Threads*
- *Waitable timers*

En Solaris y Linux también se podrán utilizar cualquier tipo de *synchronization objects*:

- *Mutex Locks*
- *Semaphores*
- *Condition variables*
- *Read-Write Locks*

Memory Management

Windows

Cada thread que la aplicación genere, creará su propio bloque de páginas adicional (su propio heap) en el espacio de dirección del proceso y al que solo él tendrá acceso. Esto lo hará para tener un manejo más eficiente de memoria y evitar el overhead generado por la sincronización entre threads. La cantidad de memoria que deberán ocupar las páginas al ser inicializadas es de 1024 Kb. No habrá restricciones en cuanto a tamaño. El tamaño máximo estará limitado por la cantidad de memoria disponible en el sistema.

Dado que el tiempo de vida de cada thread será corto, este no se ocupará de la fragmentación que se genere en el heap. Cuando el thread ya no necesite el heap este lo deberá eliminar para liberar los recursos.

En ningún momento un thread que no sea el principal utilizará la memoria del heap creado por defecto en el proceso ó heap global. Para esto se utilizarán las funciones de manejo de memoria proporcionadas por la API de Windows. No está permitido el uso de la biblioteca estándar de C para manejo de memoria (*Stdlib.h: malloc, free, etc*). Para C++ está permitido el uso de los operadores *new* y *delete* para la alocación de objetos solamente.

Solaris y Linux

No existen restricciones en cuanto al uso de memoria.

File system

Toda operación de I/O que involucre manejo de archivos en Windows deberá ser usando la Windows API correspondiente en su versión *ANSI* (en caso de existir *UNICODE*). No está permitido el uso de la biblioteca estándar de C ó de C++ para manejo archivos (*Stdio.h: fread, fwrite, fopen, etc*).

Para Linux y Solaris no existe tal restricción.

Structured Exception Handling

Windows, Solaris y Linux (sólo código C)

Queda prohibido el uso de cualquier mecanismo de exception handling, ya sea para sentencias del tipo *exception handler* (__try __except) ó *termination handlers* (__try __finally) en Windows.

Si existiese una API de Windows que permita el control de errores mediante este mecanismo, generando una excepción ante un error, se deberá forzar a que retorne un código de error para ser logueado tal como se explica en la sección de **Formato del Archivo Log**.

Código C++

Los mecanismos de *exception handling* (try, catch, throw) serán permitidos cuando el código compilado sea C++. El lenguaje ya trae mecanismos propios para manejar excepciones basadas en ANSI C++ standard.

LDAP y OpenDS

Como implementación de LDAP se utilizará OpenDS de SUN. Se encuentra disponible en la Virtual Machine de Solaris de la cátedra y se debe utilizar esta versión. El proyecto está escrito en Java y es multiplataforma. Requiere que el entorno tenga instalado un Java Runtime Environment. Se puede descargar gratuitamente de http://www.opends.org/.

Para utilizar el protocolo LDAP está permitido utilizar la API de OpenLDAP disponible en http://www.openldap.org/. Se encuentra para descargar tanto para Linux como para Solaris. También es posible la descarga con el administrador de paquetes de ambos sistemas.

La cátedra da soporte para OpenLDAP brindando un wrapper de la biblioteca original. Se puede obtener del grupo oficial de la cátedra en la sección de archivos. El proyecto se encuentra publicado en el siguiente sitio de Google Code: http://code.google.com/p/openldapwrapper/

Archivo Log y Debugging

Manejo de Errores y Excepciones

La aplicación mostrará por consola y generará en el correspondiente archivo Log el código de error y la descripción de dicho código obtenido del sistema tanto para las llamadas a la API de Windows como para las System Calls de Linux y Solaris. Para los errores pertinentes a la aplicación de deberá respetar las normas de logueo del trabajo práctico.

Algunas aclaraciones para la plataforma Windows

Dichos códigos se encuentran definidos en el header *WinError.h*. Para la obtención del *message string* a partir cierto código generado, se utilizará la función más apropiada de la API de Windows en su correspondiente versión *ANSI* (en caso de existir *UNICODE*).

Formato del archivo Log

El archivo Log deberá respetar el siguiente formato de presentación en todos los procesos que se ejecuten en cualquiera de los tres sistemas operativos.

En caso de utilizar distintos niveles detalle, el cambio entre uno u otro debe ser configurable por el usuario.

Se le recomienda al alumno registrar en este archivo los eventos más importantes de la ejecución de la aplicación, así como los valores necesarios para conocer el estado del sistema en un determinado momento. Esto es muy importante ya que en instancias finales de evaluación es probable que se haga uso de este archivo en situaciones donde la aplicación falla.

Fecha NombreProceso [PIDproceso][ThreadID]: TipoLog: Dato

Descripción

Fecha

Fecha del sistema. Deberá respetar el siguiente formato [HH:mm:ss.SSS].

Nombre Proceso

Nombre del proceso que está escribiendo en el Log.

PID Proceso

Process ID del proceso que está escribiendo en el Log.

Thread ID

ID del thread que escribe en el archivo. Opcional para el thread principal del proceso.

TipoLog

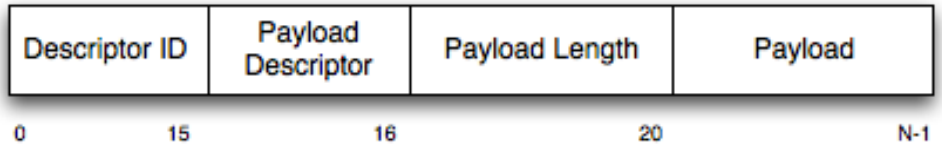
INFO, WARN, ERROR ó DEBUG nivel de detalle según lo que consideren apropiado.

Data

Descripción del evento ó cualquier información que se considere apropiada.

Protocolo IRC/IPC standard

Se utilizara un mensaje de protocolo interno. Estos son los campos mínimos que todo mensaje interno debe utilizar.



IRC/IPC Descriptor Header

Request:

Descriptor ID:

Identificador de 16 bytes único descriptor en la red.

PayloadDescriptor:

Identificador de nro de protocolo.

PayLoad Lenght:

La longitud del descriptor inmediatamente seguido del header.

Payload:

La carga de datos que se necesite pasar. Queda libre al usuario del protocolo.

Response:

Descriptor ID:

Identificador de 16 bytes correspondiente al Request.

PayloadDescriptor:

Identificador de nro de protocolo.

PayLoad Lenght:

La longitud del descriptor inmediatamente seguido del header.

Payload:

La carga de datos que se necesite pasar. Queda libre al usuario del protocolo.

Documentación

El material de soporte para poder llevar a cabo este trabajo práctico se encuentra en su mayoría en la Web. A continuación se enumeran enlaces a páginas con la información necesaria para cada plataforma.

MSDN de Microsoft *http://msdn .microsoft.com*.

Sun Microsystems Documentation http://docs.sun.com/

- **Threads y Processes**

  http://msdn.microsoft.com/en-us/library/ms686937%28VS.85%29.aspx

- **Dynamic Link Libraries**

  http://msdn.microsoft.com/en-us/library/ms682589(VS.85).aspx

- **Scheduling**

  http://msdn.microsoft.com/en-us/library/ms685096(VS.85).aspx

- **Winsocks**

  http://msdn.microsoft.com/en-us/library/ms740673%28VS.85%29.aspx

- **Memory Management**

  http://msdn.microsoft.com/en-us/library/aa366779%28VS.85%29.aspx

- **Thread Syncronization**

  http://msdn.microsoft.com/en-us/library/ms682584%28VS.85%29.aspx

- **Time Functions**

  http://msdn.microsoft.com/en-us/library/ms724962(VS.85).aspx

- **Cryptography**

  http://msdn.microsoft.com/en-us/library/aa380255(VS.85).aspx

- **Security**

  http://msdn.microsoft.com/en-us/library/cc527452.aspx

- **Standard C++ Library Reference**

  http://msdn.microsoft.com/en-us/library/cscc687y(VS.80).aspx

También se recomienda el siguiente material bibliográfico como soporte teórico:

- Programming Applications for Microsoft Windows, 4th edición de Jeffrey Richter.
- Microsoft Windows Internals, 4th Edición de Mark E. Russinovich y David A. Solomon.
- Solaris Internals: Solaris 10 and Open Solaris Kernel Architecture, Segunda Edición de Richard McDougall
- Understanding the Linux Kernel, 3rd Edición de Daniel P. Bovet, Marco Cesati

También pueden encontrar toda la documentación recomendada por nosotros, en nuestro grupo, accediendo a:

http://groups.google.com.ar/group/tp-so-frba-utn/web/links-a-tutoriales?hl=es

Información sobre cómo utilizar ***OpenLDAP SDK*** se encuentra disponible en la siguiente dirección:

http://www.openldap.org/software/man.cgi?query=ldap

Información sobre cómo utilizar ***OpenDS*** puede encontrarse en los siguientes enlaces:

https://www.opends.org/wiki/page/OpenDSUserDocumentation

Información sobre cómo utilizar ***Memcached*** puede encontrase en los siguientes enlaces:

http://www.danga.com/memcached/

La API para un cliente de ***Memcached*** las pueden encontrar en:

http://code.google.com/p/memcached/wiki/Clients

Open SSL se puede bajar desde:

http://www.openssl.org

Ejemplos clientes/servidores SSL desarrollados para Linux:

http://www.rtfm.com/openssl-examples/

Ejemplos de Berkeley DB con la API de C:

http://www.oracle.com/technology/documentation/berkeley-db/db/gsg/C/index.html

Ejemplo de MSMQ utilizando COM en C:

http://msdn.microsoft.com/en-us/library/ms811055.aspx

Un ejemplo simple (en VB) de cómo crear una Queue privada:

http://msdn.microsoft.com/en-us/library/ms704021(VS.85).aspx

MSMQ Send Message using COM

http://msdn.microsoft.com/en-us/library/ms706212(VS.85).aspx

MSMQ receive Message using COM

http://msdn.microsoft.com/en-us/library/ms705040(VS.85).aspx

Ejemplos de como enviar y recibir mensajes usando el tipo de datos Variant en COM

http://www.codersource.net/mfc_msmq_program_sample.html

libxml2 se puede descargar de:

http://xmlsoft.org/