La situación actual (contexto)

La Informática Industrial (II) en una asignatura básica que se imparte en el tercer año del Grado en Electrónica Industrial y Automática (GEIA) la Escuela de Ingenierías Industriales (EII) de la Universidad de Valladolid (UVA). Es una asignatura de 6 créditos ECTS, cuyo objetivo es el desarrollar en los estudiantes, además de las necesarias competencias generales, otras más específicas que tienen que ver con la utilización del ordenador en la industria contemporánea. Así planteado, el tema de estudio resulta inabarcable: el ordenador se utiliza a todos los niveles de la industria contemporánea desde la capa de negocios hasta el control de los elementos de campo en el proceso productivo. Entra además, directamente, la utilización de los elementos programables en la conformación del propio producto: sistemas empotrados. Se han identificado, sin embargo, los temas fundamentales de la asignatura, que pasan por la programación de sistemas reactivos, la utilización de las tareas concurrentes que de alguna manera se ejecutan "simulando" la simultaneidad del mundo real cuando se tienen ordenadores de una solo CPU o utilizando el paralelismo realmente existente en los ordenadores actuales de varios núcleos. La asignatura de II es concebida como la antesala de la asignatura de cuarto año Control y Comunicaciones Industriales (CCI) en la que se hará énfasis en los conceptos de tiempo real, de sistemas distribuidos a través de redes informáticas y en el control de sistemas continuos.

Se ha decido utilizar como lenguaje informático vehicular para la asignatura el lenguaje C++ en su versión 11. Las razones de esta decisión son las siguientes:

1. ***Lenguaje que se define como extensión multi-paradigma del lenguaje procedural C que es impartido en Fundamentos de Informática, asignatura básica que reciben todos los alumnos de la EII. El tránsito del C al C++ se puede realizar de forma "suave" introduciendo, no todo el lenguaje, sino aquellos elementos fundamentales que permitan cumplir con los objetivos de la asignatura y trabajar a un mayor nivel de abstracción.***
2. ***Entre los elementos fundamentales que permiten ese nivel de abstracción superior se encuentra las clases contenedores STL que vienen en la librería estándar. Otros elementos como el paso por referencia o por movimiento, el uso de los strings, la sobrecarga de funciones, la inicialización universal, el uso de "punteros inteligentes" y la e/s por la consola y utilización más simple de ficheros, son elementos a tener en cuenta.***
3. ***Desde la versión 11 el C++ permite el uso de la concurrencia desde el propio lenguaje, lo cual implica una ventaja notable para los contenidos concretos de la asignatura de II.***

El cambio que quisiera ver (reto)

Los retos fundamentales a vencer en la asignatura II:

1. ***Aprendizaje del C++ a partir del C haciendo énfasis en la programación a un nivel de abstracción mayor utilizando clases contendoras de la librería estándar.***

2. *Enseñar las bases de la programación concurrente, utilizando hilos (threads) dentro del propio lenguajes C++v11 o creando procesos con espacio de memoria independiente usando para ello la librería POSIX que accede a los recursos que brinda el Sistema Operativo.*

La programación es una disciplina exigente que no admite atajos, ni ambigüedades: el programa compila o no; y si compila: produce la solución correcta o no. Incluso aunque produzca la solución correcta hay otras virtudes que hacen que un programa sea adecuado, en el sentido de que tenga una estructura que lo haga mantenible por el propio autor o por sus colaboradores. La programación concurrente es aún más difícil puesto que ahora se pueden hacer pocas suposiciones sobre el orden en que ejecutan las diferentes tareas que conforman la aplicación. Esto es especialmente problemático cuando las tareas comparten datos o deben ser sincronizadas por algún otro motivo.

Se trata de trabajar a un nivel mayor de abstracción, de forma que las aplicaciones que se programen no resulten en meros ejemplos de poca entidad sino que, al contrario, tenga una relación directa con lo que se realizaría en aplicaciones de automatización realistas.

¿Cómo conseguir el cambio? (solución)

El hecho de trabajar con C++ v11 ayudaría a conseguir este cambio: tendríamos a nuestra disposición la extensa librería del C++ que brinda clases contenedoras genéricas y algoritmos muy probados. Esto nos permite trabajar a un nivel de abstracción mayor y ocuparnos de ejemplos más realistas, no teniendo que dedicar demasiados esfuerzos a crear desde cero la infraestructura informática que es siempre imprescindible. Por otra parte, las primitivas para la concurrencia que ofrece el lenguaje, también son muy útiles y de alto nivel de abstracción. Son directamente aprovechables, por otra parte, las nuevas funciones para trabajar con el tiempo que viene agrupadas en el fichero de cabecera <chrono>.

Ahora, programar y en particular, programar aplicaciones concurrentes requiere de un esfuerzo del alumno muy importante. En este contexto opinamos que resultan útiles las estrategias docentes de aprendizaje colaborativo, como por ejemplo, la pirámide o el puzle, basadas en TICs.

A modo de ejemplo, crearemos una práctica colaborativa basada en TICs que utiliza el modelo de puzle para hacer que los alumnos colaboren en definir y crear una aplicación moderadamente compleja. Cada grupo recibirán un módulo diferente de la tarea a realizar y deberán ponerse de acuerdo en la interfaz que cada tarea tendrá con las demás, para finalmente lograr compilar y poner a punto la aplicación concurrente, en este caso basada en procesos creados a nivel del sistema operativo mediante POSIX.

Medidas del éxito (evaluación)

La estrategia será un éxito si:

1. *Cada estudiante participa activamente en el proceso de elaboración del documento*

común.

2. *Como resultado de la experiencia, el estudiante comprende:*
    1. *La posibilidad de dividir un trabajo complejo entre varios equipos de trabajo.*
    2. *Lo esencial de ponerse de acuerdo la interfaz de las funciones (entradas y salidas) y documentar claramente lo que hace cada módulo, así como los mecanismos de sincronía y de comunicación entre procesos.*
3. *El profesor estará atento al desarrollo de las discusiones, para ofrecer su guía en los momentos oportunos.*

Autoría(s): rogelio.mazaeda

Publicado en: 28 Jun 2016 11:36 GMT

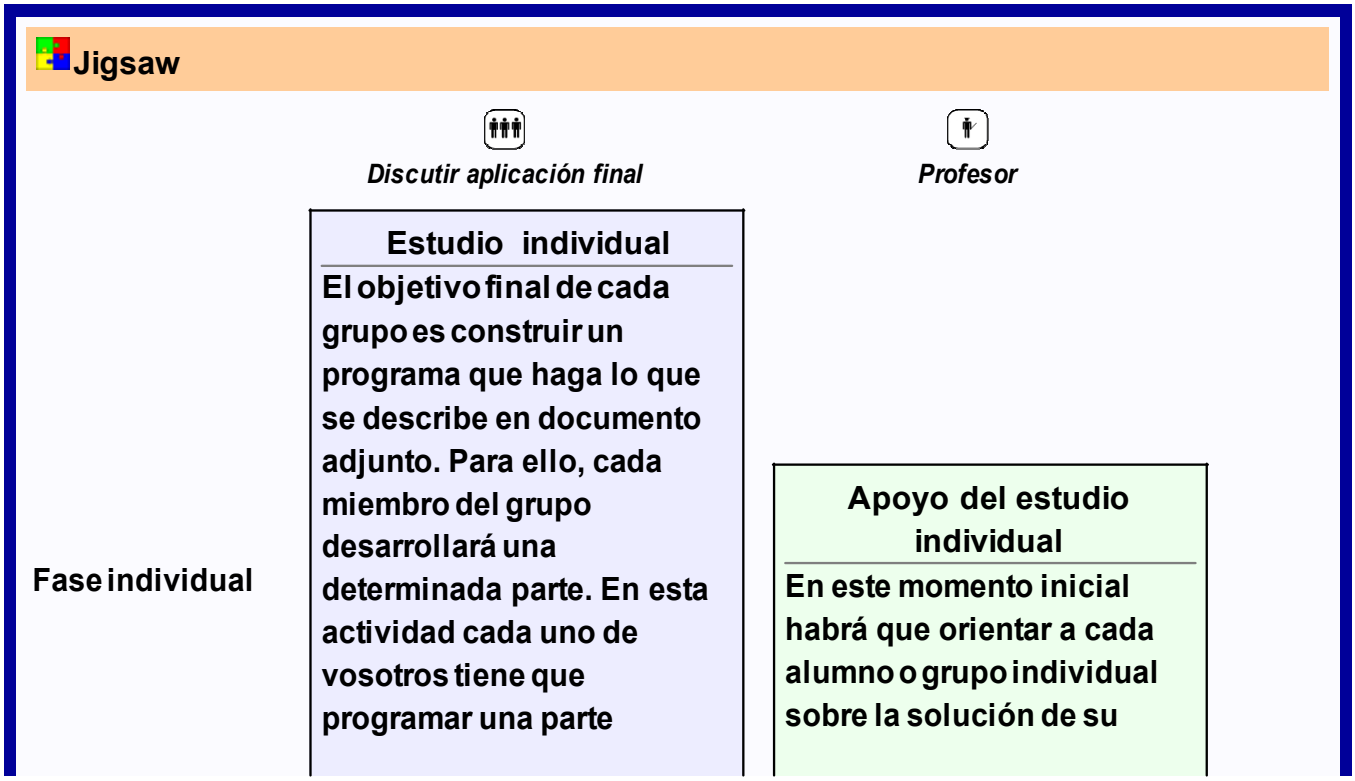
Enlace original (Inicio de sesión requerido): http://ilde.upf.edu/uva/pg/lds/view/11557/

General information:

Title: Autoría (FI 2016/2017)
Prerrequisites: Dominio de los elementos básicos del lenguaje C y del control automático.

- **Entender las ventajas de la programación modular: Muchas veces el alumno no percibe de forma inmediata las ventajas del uso de funciones. Prefiere, por ejemplo, hacerlo todo en la función principal. Una práctica colaborativa en la que cada experto se dedique a programar una función o módulo específico de un programa de complejidad moderada y que para ello tenga que colaborar con los otros programadores en definir la interfaz entre los módulos puede ser muy útil en este contexto.**
- **Entender descripción: A partir de la descripción de las tareas que debe realizar un módulo, ser capaz de analizar posibles alternativas para su solución en el lenguaje C++**
- **Conocer elementos de la programación recurrente: Conocer el concepto de hilos o procesos. Saber cómo crearlos y utilizarlos tanto en forma nativa al lenguaje (hilos en C++v11) como utilizando el estándar POSIX a nivel del sistema operativo.**
- **Conocer tratamiento de las secciones críticas: Conocer la manera de garantizar la validez de los recursos compartidos por varios procesos concurrentes.**
- **Conocer elementos de sincronización: Conocer como sincronizar el funcionamiento recurrente de varias tareas.**
- **Diseñar estructuras de datos utilizando elemento de la biblioteca STL de C++11: Trabajar a un nivel de abstracción mayor, utilizando elementos STL tales como vectores o listas.**
- **Utilizar el patrón RAII para acceso a elementos compartidos: Utilizar el conocido patrón RAII (*Resource adquisition is initialization*) a la hora de acceder a los recursos compartidos, como una buena práctica de programación que mejora la fiabilidad de la solución buscada.**

Learning activity flow:



	diferente del sistema a desarrollar: - Parte 1: Desarrollo proceso control. - Parte 2: Desarrollo proceso lista. - Parte 3: Desarrollo del proceso sensor.	tarea particular.
	expertos en funciones	*Profesor*
Fase de expertos	**Fase expertos** Los alumnos a los que se le ha asignado cada función, deberán discutir las posibles alternativas para la solución e identificar los datos de entrada necesarios y la solución a brindar. Identificar las posibles condiciones de carrera y su solución. Existencia y solución de posibles bloqueos.	**Apoyo a fase de experto** El profesor apoyará cada grupo de expertos tratando de dirigir la discusión hacia los recursos que cada función del programa (a cargo de cada grupo de expertos) requiere del resto de las funciones y que debe entregar. Esta necesidad del interfaz estará por supuesto determinada por el procesamiento específico que cada función deba realizar. En este punto del proceso, cada grupo de expertos debe ser consciente de que habrán detalles que deberán ser pulidos en la fase posterior, tal como el orden en se reciben los parámetros o la forma concreta en que se entrega el resultado. De manera que el profesor cuidará que la discusión se mantenga abierta en estos temas, contemplando posibles variantes.
	Discutir aplicación final	*Profesor*
	Discusión global El experto de cada función conoce las necesidades que la misma tiene en cuanto a datos de entrada y	

Fase de jigsaw	el resultado que debe entregar. Todos los expertos deberán ponerse de acuerdo sobre los tipos de datos concretos a intercambiar y orden de los parámetros. Deben elegir los tipos de datos de la aplicación que van a ser intercambiados entre las distintas funciones.	**Apoyo de la discusión** El profesor estará atento a la discusión de forma que se cumplan los objetivos. Deberá recalcar que el objetivo es poner en común la interfaz entre las funciones y los tipos de datos a intercambiar, identificar los elementos es que es necesaria la sincronización, evitar condiciones de carrera y *deadlocks*.
	Propuesta de solución Con los prototipos de las funciones definidos, cada experto deberá implementar en lenguaje C las funciones a su cargo.	

Se pide simular un sistema en el que un conjunto de sensores trasmiten sus lecturas a un proceso de control. Este a su vez gestiona el almacenamiento de estas en una lista.

Concretamente el sistema está formado por los siguientes elementos (véase la figura):

Un proceso de control que se comunica a lo sumo con MAX_SENSORES procesos sensor activos e intercambia información con ellos. MAX_SENSORES es consignado por el usuario mediante la línea de comandos, al lanzar el proceso de control. Este proceso de control lleva a cabo la siguientes tareas:

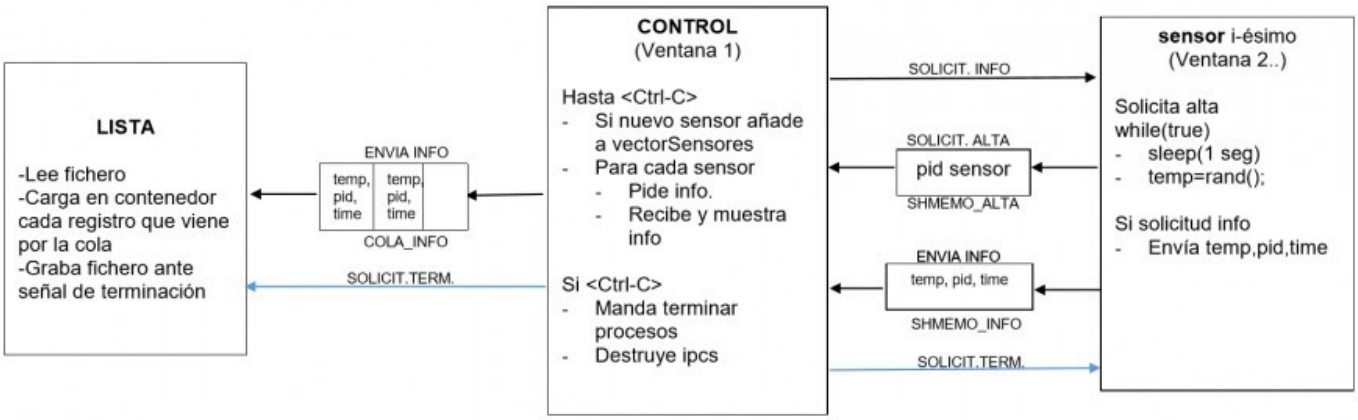
- **Detecta los sensores que se incorporan al sistema, a los que identifica por su pid.**
- **Solicita periódicamente la temperatura leída. Si alguno de los sensores no responden al cabo de un cierto tiempo de espera (timeout) el proceso de control eliminará dicho sensor del sistema.**
- **Recibe, a través de memoria compartida, la información enviada por cada sensor (temperatura leída, pid del sensor e instante de lectura timestamp) y solicita al proceso lista guardar esta información en un contenedor.**
- **El proceso de control termina cuando se pulsa un retorno de Ctrl+C. Antes de finalizar debe enviar señal de terminación a todos los procesos conectados (sensores y proceso lista) y destruir todos los mecanismos de comunicación y sincronización creados.**

Una serie de procesos sensor idénticos. Cada sensor:

- **Deberá solicitar el alta al proceso control enviándole su pid a través de shmemo_alta. Como este permite solamente la conexión de MAX_SENSORES, si se intenta conectar alguno más, los que excedan este número quedarán bloqueados hasta que otros sensores activos sean dados de baja.**
- **Los sensores envían al proceso de control, a petición de él, información sobre la temperatura leída. Para ello emplearán la memoria compartida shmemo_sensor.**

Un proceso lista que gestiona un contenedor con todas las lecturas de los sensores:

- **Al arrancar este proceso carga el contendor con las lecturas de la sesión anterior que figuran en un fichero.**
- **Queda a la espera de la informaciÃ³n que reciba del proceso de control por una cola de mensajes. Introduce en el contenedor los registros de los sensores con los campos temp, pid y tiempo que llegan por la cola de mensajes.**
- **Ante la llegada de la solicitud de terminación, antes de finalizar guardará la información del contenedor en un fichero para que pueda ser cargada en la siguiente sesión.**



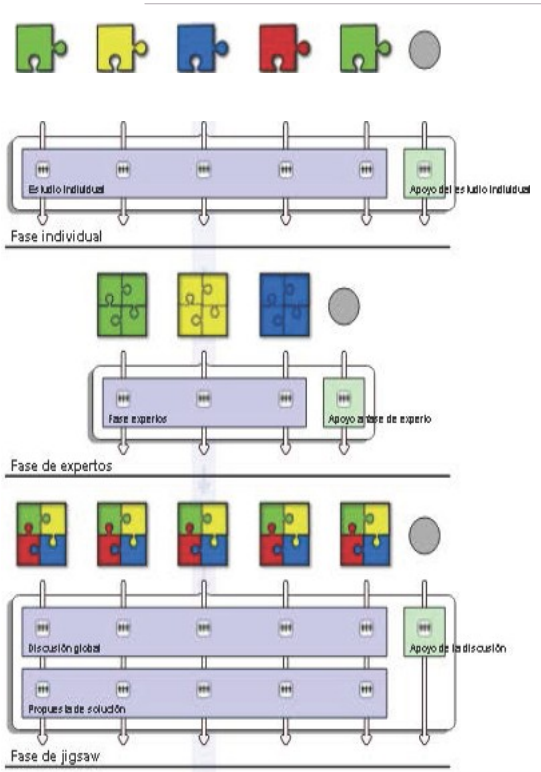
Autoría(s): rogelio.mazaeda

Publicado en: 28 Jun 2016 12:04 GMT

Enlace original (Inicio de sesión requerido): http://ilde.upf.edu/uva/pg/lds/vieweditor/11560/

General Flujo de aprendizaje Recursos Resumen

- Estructura
  - main (Jigsaw)
    - <:> Fase individual
    - <:> Fase de expertos
    - <) Fase de jigsaw



Documento de soporte A ProcesoControl A ProcesoSensor A Procesolista A + Añadir un documento **WebCollage**