TEMA 1. FUNDAMENTOS DE LA CONCURRENCIA

Ingeniería en Informática

Contenidos

- Introducción
- Beneficios de la programación concurrente
- Concurrencia y arquitecturas hardware
- Especificación de ejecución concurrente
- Características de los sistemas concurrentes
- Problemas inherentes a la programación concurrente
- Verificación de programas concurrentes

Bibliografía

- Programción Concurrente
  - J. Tomás, M. Garrido, F. Sánchez, A. Quesada, 2003
  - Capítulo 1 y 2
- Principles of Concurrent and Distributed Programming
  - M. Ben-Ari. Prentice Hall, 1990
  - Capítulo 1 y 2
- Concurrent Programming
  - A. Burns, G Davis. Addison-Wesley, 1993
  - Capítulo 1

Contenidos

- **Introducción**
- Beneficios de la programación concurrente
- Concurrencia y arquitecturas hardware
- Especificación de ejecución concurrente
- Características de los sistemas concurrentes
- Problemas inherentes a la programación concurrente
- Verificación de programas concurrentes

¿Qué es la concurrencia?

- Definición Real Academia Española:

<<Acaecimiento o concurso de varios sucesos en un mismo tiempo>>

- Una forma de ver la concurrencia es como un conjunto de actividades que se desarrollan de forma simultánea
- En informática, cada una de esas actividades se suele llamar **proceso**

Concurrencia

- En Informática, se habla de concurrencia cuando hay una *existencia simultánea de varios procesos en ejecución*
- Dos procesos serán concurrentes cuando la primera instrucción de uno de ellos se ejecuta después de la primera instrucción del otro y antes de la última

Concurrencia

- Ojo, *existencia simultánea* no implica *ejecución simultánea*
  - Dependerá del hardware subyacente
- El paralelismo es un caso particular de la concurrencia
- Se habla de *paralelismo* cuando ocurre la ejecución simultánea de instrucciones:
  - arquitecturas paralelas
  - procesamiento paralelo
  - algoritmos paralelos
  - programación paralela

¿Dónde se encuentra la concurrencia?

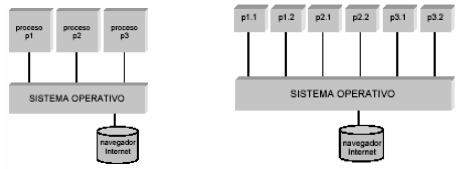
- En la Naturaleza (el problema que se modela)
  - ¿ejemplos?
- En el *hardware* (la herramienta para solucionar el problema):
  - ejecución paralela de instrucciones
  - funcionamiento paralelo de los periféricos
  - procesadores múltiples
  - sistemas distribuidos

Concurrencia inherente o potencial

- Sistemas inherentemente concurrentes:
  - el entorno con el que interactúan, o el entorno que modelan tiene forzosamente actividades simultáneas
  - p.ej. red de cajeros automáticos
- Sistemas potencialmente concurrentes:
  - no es estrictamente necesario que haya concurrencia, pero se puede sacar partido de ella
  - p.ej. para aumentar la velocidad de ejecución

Programa y Proceso

- Definición extendida: un proceso es un programa en ejecución
- o mejor “una actividad asíncrona susceptible de ser asignada a un procesador”



Procesos

- Independientes
  - P.ej. dos procesos de dos usuarios diferentes
- Cooperantes
  - P. ej. el navegador del esquema anterior
- Estas tareas de colaboración y competencia exigen mecanismos de **comunicación** y **sincronización** entre procesos

Programación Concurrente

Disciplina que se encarga del estudio de las notaciones que permiten especificar la ejecución concurrente de las acciones de un programa, así como las técnicas para resolver los problemas inherentes a la ejecución concurrente (comunicación y sincronización)

Programación Concurrente

- Tradicionalmente estuvo asociada al mundo de los Sistemas Operativos
- Primeros programas concurrentes: SO
  - Evolución plataformas hardware
  - Diferentes partes del SO en ejecución sin un orden predecible y compartiendo variables => nuevos problemas
  - Soluciones específicas
- Tres hitos importantes marcan su evolución:
  - Hilo
  - Aparición de lenguajes de alto nivel que da soporte a la P.C.
  - La aparición de Internet

Programación concurrente

- EL trabajar con procesos concurrentes añade complejidad a la tarea de programar
- ¿cuáles son entonces los beneficios que aporta la programación concurrente?

Contenidos

- Introducción
- **Beneficios de la programación concurrente**
- Concurrencia y arquitecturas hardware
- Especificación de ejecución concurrente
- Características de los sistemas concurrentes
- Problemas inherentes a la programación concurrente
- Verificación de programas concurrentes

Beneficios de la programación concurrente

- Mejor aprovechamiento de la CPU
- Velocidad de ejecución
- Solución de problemas de naturaleza concurrente
  - Sistemas de control
  - Tecnologías web
  - Aplicaciones basadas en interfaces de usuarios
  - Simulación
  - SGDB

Contenidos

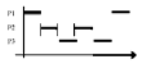
- Introducción
- Beneficios de la programación concurrente
- **Concurrencia y arquitecturas hardware**
- Especificación de ejecución concurrente
- Características de los sistemas concurrentes
- Problemas inherentes a la programación concurrente
- Verificación de programas concurrentes

Concurrencia y arquitecturas hardware

- Arquitecturas que soportan la programación concurrente
  - Sistemas con un solo procesador (sistemas monoprocesador)
  - Sistemas con más de un procesador (sistemas multiprocesador)

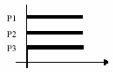
Sistemas monoprocesador

- No hay paralelismo. Los procesos se reparten el procesador: entrelazado (interleaving)
- ¿Quién planifica los procesos?
  - el sistema operativo
  - el propio ejecutable (gracias al compilador) -> *runtime scheduler (RTSS)*
- Todos los procesos comparten la misma memoria
- Forma "natural" de sincronizar y comunicar procesos
  - Variables compartidas



Sistemas multiprocesador o sistemas paralelos

- Más de un procesador => permite que exista un paralelismo real entre los procesos



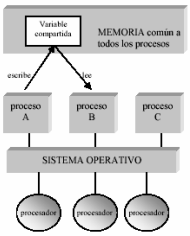
- Clasificación sistemas multiprocesador
  - estrechamente acoplados (multiprocesadores)
  - débilmente acoplados (sistemas distribuidos)

Sistemas estrechamente acoplados

- Normalmente llamados **multiprocesadores**
- Los procesadores comparten memoria y reloj
  - Puede ser más o menos difícil acceder a la memoria de otro procesador
- Ventaja:
  - aumento de velocidad de procesamiento con bajo coste
- Inconveniente:
  - Escalable sólo hasta decenas o centenares de procesadores

Sistemas estrechamente acoplados

- La sincronización y comunicación entre procesos se suele hacer mediante variables compartidas



Sistemas débilmente acoplados o sistemas distribuidos

- Múltiples procesadores conectados mediante una red
- Los procesadores no comparten memoria ni reloj
- Los sistemas conectados pueden ser de cualquier tipo
- Escalable hasta millones de procesadores (ej. Internet)

Sistemas distribuidos: ventajas

- compartición de recursos dispersos
- ayuda al trabajo cooperativo de equipos humanos
- aumento de velocidad de ejecución
- escalabilidad ilimitada
- aumento de fiabilidad:
  - tolerancia a fallos (fault tolerance)
  - alta disponibilidad (availability)

Sistemas distribuidos

- La forma natural de comunicar y sincronizar procesos es mediante el uso de paso de mensajes

Contenidos

- Introducción
- Beneficios de la programación concurrente
- Concurrencia y arquitecturas hardware
- **Especificación de ejecución concurrente**
- Características de los sistemas concurrentes
- Problemas inherentes a la programación concurrente
- Verificación de programas concurrentes

Especificación de ejecución concurrente

- ¿Qué se puede ejecutar concurrentemente?
  - Condiciones de Bernstein (1966)
- ¿Cómo especificarlo?
  - Lenguaje concurrente

Condiciones de Bernstein

- Sea
  - $L(S_k)=\{a_1,a_2,...,a_n\}$ el conjunto de lectura del conjunto de instrucciones S_k formado por todas las variables cuyos valores son referenciados (se leen) durante la ejecución de las instrucciones en S_k
  - $E(S_k)=\{b_1,b_2,...,b_n\}$ el conjunto de escritura del conjunto de instrucciones S_k formado por todas las variables cuyos valores son actualizados (se escriben) durante la ejecución de las instrucciones en S_k

Condiciones de Bernstein

- Para que dos conjuntos de instrucciones S_i y S_j se puedan ejecutar concurrentemente, se tiene que cumplir que:
  - 1.- $L(S_i) \cap L(S_j) = \emptyset$
  - 2.- $E(S_i) \cap L(S_j) = \emptyset$
  - 3.- $E(S_i) \cap E(S_j) = \emptyset$

¿Cómo especificar la concurrencia?

- Técnicas para producir actividades concurrentes en el computador
  - De forma manual
    - Trabajar directamente sobre el hardware
    - Usar llamadas al sistema o bibliotecas de software (ejs. PVM, pthreads)
    - Expresarla en un lenguaje de alto nivel
  - De forma automática
    - El sistema operativo se encarga automáticamente (ej. multiprogramación)
    - El compilador detecta la *concurrencia implícita* en nuestros programas secuenciales

Lenguajes de alto nivel

- Los LAN permiten programar en un nivel más cercano al *ámbito del problema* (*problem domain*) por medio de la **abstracción**
- Además, los lenguajes y el S.O. proporcionan herramientas para usar con más comodidad los recursos del hardware.

Lenguajes de alto nivel

- A lo largo de la historia, se han inventado abstracciones en los LAN que han resultado muy útiles para la comunidad informática:
  - abstracción de expresiones (FORTRAN...)
  - abstracción del flujo de control: programación estructurada secuencial (Algol...)
  - abstracción de la lógica y el álgebra (Lisp, Prolog...)
  - abstracción de datos (Algol, Pascal...)
  - modelado de objetos (Smalltalk, C++...)
  - **¿abstracción de la concurrencia?**

Abstracción de la concurrencia

- Nuestro programa expresa acciones concurrentes (procesos o hilos), pero éstas no tienen por qué ejecutarse en paralelo
- Cada proceso concurrente se ejecuta sobre un **procesador virtual**
- El compilador y el s.o. serán responsables de ejecutar nuestros procesos como consideren más oportuno
- Nos deben dar igual las velocidades relativas de los procesadores virtuales

Lenguajes concurrentes

- Aquellos que incorporan características que permiten expresar la concurrencia directamente, sin recurrir a servicios del s.o., bibliotecas, etc.
- Normalmente incluyen mecanismos de sincronización y comunicación entre procesos
- Ejemplos: Ada, Java, SR, Occam, PARLOG...

¿Cómo expresar la concurrencia? Ejemplos

- Sentencia concurrente:
```
cobegin
  P; Q; R
coend;
```
- Sentencia concurrente múltiple:
```
forall i:=1 to 1000 do
  P(i);
```
- Tuberías (unix):
```
grep palabra | sort | lpr
```
- Instrucciones vectoriales:
```
type vector is
  array(1..10) of int;
var a,b,c : vector;
a := b*c + 2*a;
```
- Objetos que representan procesos:
```
task A is begin P; end;
task B is begin Q; end;
task C is begin R; end;
```
- Notación gráfica: Grafos de precedencia

Contenidos

Características de los sistemas concurrentes

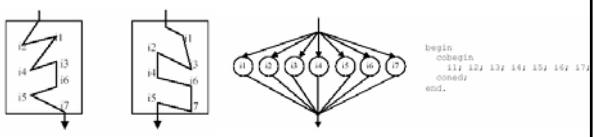
- Orden de ejecución de las instrucciones
- Indeterminismo

Orden de ejecución de las instrucciones

- La programación secuencial define un **orden total** de las instrucciones
  - Ante un conjunto de datos de entrada el flujo de ejecución es siempre el mismo

Orden de ejecución de las instrucciones

- Un programa concurrente define un **orden parcial** de ejecución
  - Ante un conjunto de datos de entrada no se puede saber cual va a ser el flujo de ejecución



Indeterminismo

- El orden parcial implica el **no determinismo** de los programas concurrentes
  - Es decir, puede producir diferentes resultados cuando se ejecuta repetidamente sobre el mismo conjunto de datos de entrada

Indeterminismo: ejemplo

- ¿Qué valor tendrá la x al terminar el programa?

```
program Incognita;
  var x: integer;

process P1;
  var i: integer;
begin
  for i:=1 to 5 do x:=x+1;
end;

process P2;
  var j: integer;
begin
  for j:=1 to 5 do x:=x+1
end;
```

```
begin
  x:=0;
  cobegin
    P1;
    P2;
  coend;
end.
```


No determinismo

- El no determinismo es una propiedad inherente a la concurrencia
- Por culpa del no determinismo, es más difícil analizar y verificar un algoritmo concurrente
- Ojo, que existan varias posibilidades de salida NO significa **necesariamente** que un programa concurrente sea incorrecto

Contenidos

- Introducción
- Beneficios de la programación concurrente
- Concurrencia y arquitecturas hardware
- Especificación de ejecución concurrente
- Características de los sistemas concurrentes
- **Problemas inherentes a la programación concurrente**
- Verificación de programas concurrentes

Problemas inherentes a la programación concurrente

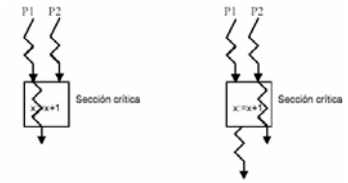
- Verificación
- Exclusión mutua
- Condición de sincronización

Exclusión mutua

- Problema derivado de la abstracción en los lenguajes de alto nivel
  - Una instrucción de alto nivel se convierte en un conjunto de instrucciones máquina
  - Son las que realmente se ejecutan concurrentemente
  - En el paradigma secuencial este hecho carece de importancia pero en ejecuciones concurrentes de instrucciones el resultado puede ser incorrecto

Exclusión mutua

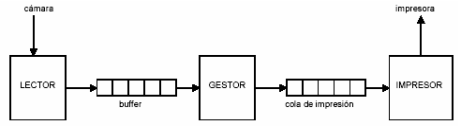
- Sección crítica: porción de código con variables compartidas y que debe ejecutarse en exclusión mutua
  - Los lenguajes concurrentes deben proporcionar herramientas para resolver este tipo de problemas



Condición de sincronización

- Muchas posibilidades diferentes en cuanto al orden de ejecución
- Sin embargo, pueden ser necesarias ciertas restricciones en el orden de ejecución

Condición de sincronización




```
process lector;
begin
 repeat
  captura imagen;
  almacena en buffer;
 forever
end;
```

```
process gestor;
begin
 repeat
  coge imagen de buffer;
  trata imagen;
  almacena imagen en cola;
 forever
end;
```

```
process impresor;
begin
 repeat
  coge imagen de cola;
  imprime imagen;
 forever
end;
```

Contenidos

- Introducción
- Beneficios de la programación concurrente
- Concurrencia y arquitecturas hardware
- Especificación de ejecución concurrente
- Características de los sistemas concurrentes
- Problemas inherentes a la programación concurrente
- **Verificación de programas concurrentes**

Verificación de programas concurrentes

- ¿Cuándo un programa (secuencial) se considera correcto?
  - **Parcialmente correcto.** Dadas unas precondiciones correctas, si el programa termina se cumplen las postcondiciones previstas
  - **Totalmente correcto.** Dadas unas precondiciones correctas, el programa termina y se cumplen las postcondiciones

Peculiaridades de los programas concurrentes

- Los programas concurrentes pueden no terminar nunca y al mismo tiempo ser correctos
- Un pr.c. puede tener múltiples secuencias de ejecución
- Cuando se dice que un pr.c. es correcto, se entiende que se refiere a *todas* sus posibles secuencias de ejecución

Verificación de programas concurrentes

- La corrección de programas concurrentes se puede determinar en base al cumplimiento de una serie de propiedades
- Dos tipos de propiedades útiles en los sistemas concurrentes
  - **Propiedades de seguridad (safety)**
    - Son aquellas que aseguran que nada malo va a pasar durante la ejecución del programa
  - **Propiedades de viveza (liveness)**
    - Son aquellas que aseguran que algo bueno pasará eventualmente durante la ejecución del programa

Verificación de programas concurrentes

- **Propiedades de seguridad (safety)**
  - Exclusión mutua
  - Condición de sincronización
  - Interbloqueo (pasivo) - deadlock
- **Propiedades de viveza (liveness)**
  - Interbloqueo (activo) – livelock
  - Inanición - starvation

Análisis de algoritmos concurrentes

- Usar invariantes y lógica proposicional
- Usar métodos inductivos
- Usar historias de ejecución (a->b)
- Usar predicados posicionales: at(l), in(l), after(l)
- Usar *lógica temporal*
- ...