Lenguajes de programación

Programación Concurrente y Paralela

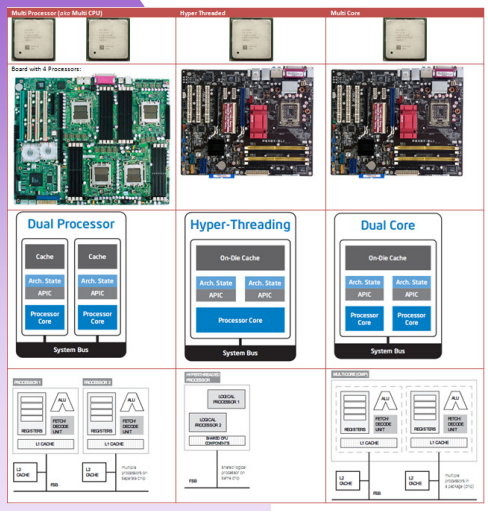
Cómputo Concurrente

- Forma de cómputo en el cual varios procedimientos son ejecutados durante períodos de tiempo traslapados en lugar de ejecutarse secuencialmente.
- La ejecución puede llevarse a cabo en un procesador físico (CPU) o en varios.

Cómputo Paralelo

- Forma de cómputo en el cual varios procedimientos son ejecutados simultáneamente en:
  - sistemas multiprocesador (*multiprocessor* - **SMP**) o
  - sistemas multinúcleo (*multicore* - **CMP**) o
  - sistemas Multihilo (*multithreaded* - **SMT**).

Hardware de Cómputo Paralelo



- **Multiprocesador** – varios procesadores físicos (CPUs) conectados por memoria o red.
- **Multinúcleo** – varios procesadores físicos dentro del mismo chip.
- **Multihilo** – simula varios procesadores lógicos dentro de un único procesador físico.

Procesos e hilos

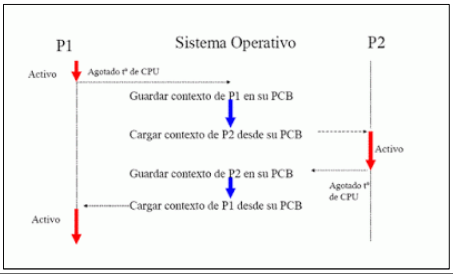
- Un **proceso** es una instancia de un programa en ejecución
  - Pueden existir varios procesos ejecutando un mismo programa, pero cada uno es un proceso distinto, con su propia representación (PCB)
- Un **hilo** (*thread*) de ejecución es la secuencia más pequeña de instrucciones programadas que pueden ser manejadas independientemente por un planificador (*scheduler*).
  - El planificador es el que decide cómo dar acceso a los recursos de un sistema (tiempo de procesador, ancho de banda de comunicación, etc.)

Procesos

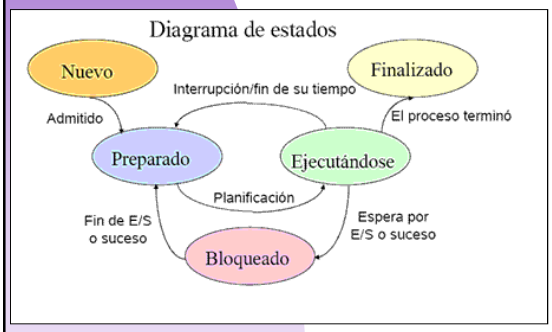
- Un proceso consta al menos de:
  - El código del programa.
  - Los datos del programa.
  - Una pila de ejecución.
  - El PC indicando la próxima instrucción.
  - Un conjunto de registros de propósito general con los valores actuales.
  - Un conjunto de recursos del SO (memoria, archivos abiertos, etc.)
  - Para la planificación de la CPU lo importante son los procesos, no los programas.

Procesos concurrentes

- Pueden ser ejecutados en un solo núcleo intercalando los pasos de ejecución de cada proceso mediante segmentos de tiempo (**multitarea apropiativa** – *preemptive multitasking*).
- Solamente se ejecuta un proceso a la vez, y si no se completa durante el segmento de tiempo, es pausado, y otro proceso inicia o resume su ejecución, para luego resumir el proceso original.



Estados y Transiciones



- Cada proceso tiene un estado de ejecución que indica lo que está haciendo actualmente, p. ejemplo:
  - **Nuevo** – El proceso se está creando.
  - **Ejecutándose** - ejecutando instrucciones en la CPU.
  - **Preparado o ejecutable** - en espera de la CPU.
  - **Bloqueado** - esperando por un suceso.
  - **Terminado** – El proceso terminó su ejecución.
- Durante su vida en el sistema, un proceso va pasando de un estado a otro.

PCB (Bloque de Control de Proceso)

- Estructura de datos que representa al proceso, es decir, que contiene la información asociada con cada proceso:
  - Estado actual del proceso.
  - Valores de los registros de la CPU.
  - Información de planificación.
  - Información para la administración de memoria.
  - Información del estado de las E/S.
  - Información de contabilidad o estadística.
  - Suceso por el cual el proceso está bloqueado.

PCB's y Colas de Estados

- El sistema de cómputo concurrente mantiene una colección de colas que representan el estado de todos los procesos en el sistema.
  - Típicamente hay una cola por estado.
  - Cada PCB esta en una cola de estado acorde a su estado actual.
  - Conforme un proceso cambia de estado, su PCB es retirado de una cola y agregado en otra.

Cambio de Contexto

- Cuando un proceso esta ejecutándose, su PC, puntero a pila, registros, etc., están cargados en la CPU (es decir, los registros hardware contienen los valores actuales).
- Cuando se detiene un proceso ejecutándose, salva los valores actuales de estos registros (**contexto**) en el PCB de ese proceso.
- La acción de conmutar la CPU de un proceso a otro se denomina **cambio de contexto**.
- Los sistemas de tiempo compartido realizan de 100 a 1000 cambios de contexto por segundo.
- Este trabajo es **sobrecarga**.

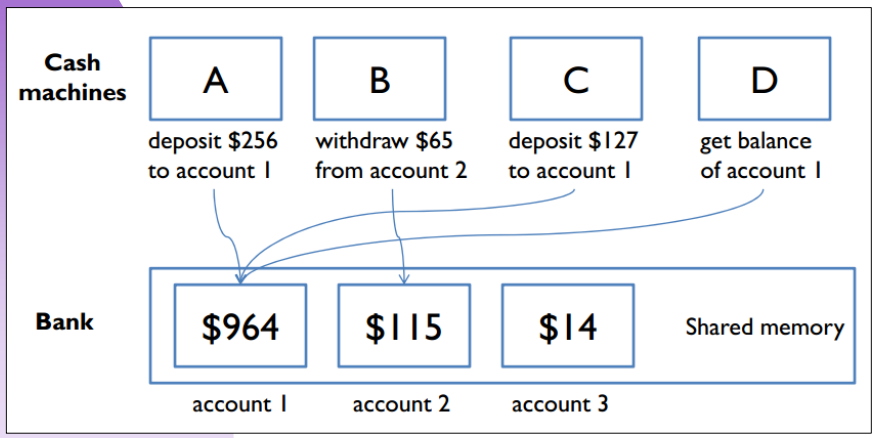
Modelo de Programación

- Describe la **forma de interacción y comunicación concurrente**
- En algunos sistemas de cómputo concurrente ha sido ocultada del programador.
- En otros debe ser manejada explícitamente.
- La comunicación explícita puede dividirse en 2 clases:
  - **Memoria compartida**
  - **Paso de mensajes**

Memoria Compartida

- Los componentes concurrentes se comunican mediante la alteración del contenido de ubicaciones de memoria compartida *(Java o C#).*
- Este estilo de programación concurrente usualmente requiere la aplicación de alguna forma de bloqueo (**mutexes**, **semáforos** o **monitores**) para la coordinación (**sincronización**) entre procesos o hilos.
- Un programa que implementa adecuadamente cualquiera de estos mecanismos se dice que tiene seguridad en hilos (**thread-safe**).

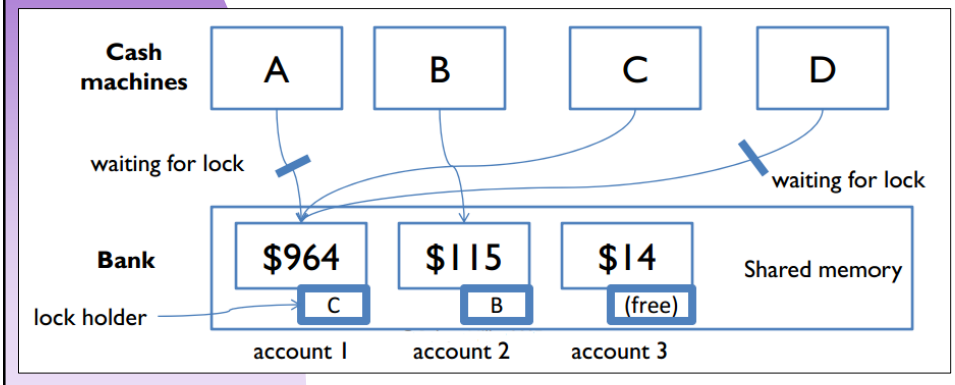
Ejemplo: Uso de ATMs



Peligros de la Ejecución Concurrente

- **Condición de carrera:** Cuando varios procesos acceden al mismo tiempo y cambian el estado de un recurso compartido (por ejemplo una variable), obteniendo de esta forma un valor que depende del orden de su ejecución.
- Si no se sincronizan correctamente, puede producirse una corrupción de datos.
- **Ejemplo**: Acceso concurrente mediante ATMs de varios clientes a cuentas de banco compartidas.

Posible solución: uso de candados



Más peligros…

- El **bloqueo mutuo** (*deadlock*): es el bloqueo permanente de un conjunto de procesos o hilos de ejecución en un sistema concurrente que compiten por recursos del sistema o bien se comunican entre ellos.
- No existe una solución general para los interbloqueos.
- **Ejemplo**: 2 niños que quieren disparar un arco, pero uno agarró el arco y el otro la flecha, y se quedan esperando a que el otro deje lo que agarró.

Más peligros…

- ***Livelock:*** es similar a un *deadlock*, excepto que el estado de los dos procesos envueltos en el *livelock* constantemente cambia con respecto al otro.
- **Ejemplo**: 2 personas en un pasillo estrecho que se bloquean mutuamente y se mueven al unísono para dejar pasar al otro, manteniendo el bloqueo.

Más peligros…

- **Inanición** (*starvation*) : cuando a un proceso o un hilo de ejecución se le deniega siempre el acceso a un recurso compartido que requiere para terminar su tarea.
- **Ejemplo**: Problema de los filósofos cenando

Conceptos a considerar

- **Exclusión mutua (EM)** se refiere al requerimiento de asegurar que ningún par de procesos se encuentren en su sección crítica al mismo tiempo para prevenir condiciones de carrera.
- Una **sección crítica** es una pieza de código que accesa un recurso compartido (estructura de datos o dispositivo) que no debe ser accesada concurrentemente por más de un hilo de ejecución.

Herramientas para EM

- Los **mutexes** son banderas que se utilizan para este indicar cuando un recurso.
- Un **semáforo** es una variable o tipo de dato abstracto que registra cuantas unidades de un recurso particular están disponibles, acoplado con operaciones para su ajuste seguro (sin condiciones de carrera) y es usado para controlar el acceso a un recurso común por varios procesos concurrentes.

Herramientas para EM

- Un **monitor** es un mecanismo de sincronización que permite que los hilos se ejecuten con exclusión mutua y tengan la habilidad para esperar (bloquearse) hasta que cierta condición se haga verdadera.
- Un monitor se compone de un **mutex** y **variables de condición** (contenedores de hilos que esperan que una condición se cumpla).

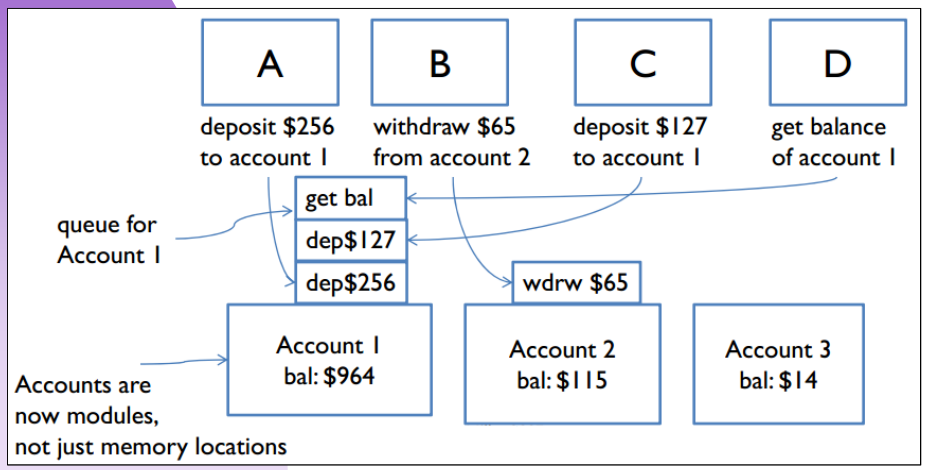
Paso de Mensajes

- Los componentes concurrentes se comunican mediante el intercambio de mensajes (*Erlang, Go y occam*).
- El intercambio de mensajes se puede efectuar **asíncronamente**, o puede usar estilo **síncrono** "*rendezvous*" en el cual el que envía bloquea hasta que se recibe el mensaje.
- El paso asincrónico de mensajes puede ser confiable o no confiable (*send and pray*).
- Esta forma de comunicación tiende a ser **más fácil de razonarse** que la de memoria compartida, y típicamente se considera una forma **más robusta** de programación concurrente.

Interacción mediante Paso de Mensajes

- Los mensajes recibidos (solicitudes) son puestos en una cola para ser manejados uno a la vez.
- El que envía no para de trabajar mientras espera una respuesta a su mensaje y así sigue atendiendo a los mensajes de su propia cola.
- Eventualmente regresan las respuestas mediante otros mensajes.

Ejemplo: Uso de ATMs



Peligros en Paso de Mensajes

- No elimina las condiciones de carrera.
  - **Ejemplo**: mandar mensajes de retiro de dinero sin checar si hay suficientes fondos.
- Tampoco elimina *deadlocks*.
  - **Ejemplo**: dos procesos se quedan esperando respuestas a mensajes para responder mensajes.

Lenguaje Concurrente

- Utilizaremos un lenguaje de programación concurrente basado en el modelo de paso de mensajes
- Instalar el lenguaje de programación Erlang
  http://www.erlang.org/