Sincronización de procesos

Proceso

- *Unidad de trabajo del sistema*
  - *Un proceso por actividad*
    - *Ejms:*
      - *Procesos de sistema:*
        - *Calendarización*
        - *E/S*
        - *Sistema de archivos*
      - *Procesos de usuario*
        - *Navegador web*
        - *Editor de textos*

Proceso

- *Dinámico*
  - *Su estado va cambiando*
  - *Corresponde a la ejecución de uno o más programas/rutinas*

Proceso

- *Dinámico*
  - *Recursos asociados*
    - *Identificador de procesos (PID)*
    - *Memoria*
    - *Archivos*
    - *Sockets*
    - *Apuntador de instrucción actual*
  - *Administrados por el S.O.*
    - *En tiempo de creación*
    - *En tiempo de ejecución*

Ejemplo de top

Programa/rutina

- *Conjunto de instrucciones y datos que describen los pasos de una tarea*
- *Ejemplos:*
  - *ordenar arreglos: qsort(libc)*
  - *limpiar pantalla: /bin/clear*
  - *Editor: c:\\Windows\Notepad.exe*

Programa/rutina

- *Conjunto de instrucciones y datos que describen los pasos de una tarea*
  - *Estático*
  - ***En lenguaje de máquina***

Procesador

- *<u>Agente</u> que ejecuta las intrucciones de un programa, para poder llevar a cabo un proceso*
  - *El proceso “corre” en el procesador*
  - *El procesador “corre” el proceso*

Procesador

- *El programa asociado a un proceso no necesariamente está en SW*
  - *Ejemplo: Rutinas de un controlador USB (firmware)*
    - *Un procesador capaz de ejecutar un sólo tipo de proceso*

Aplicación

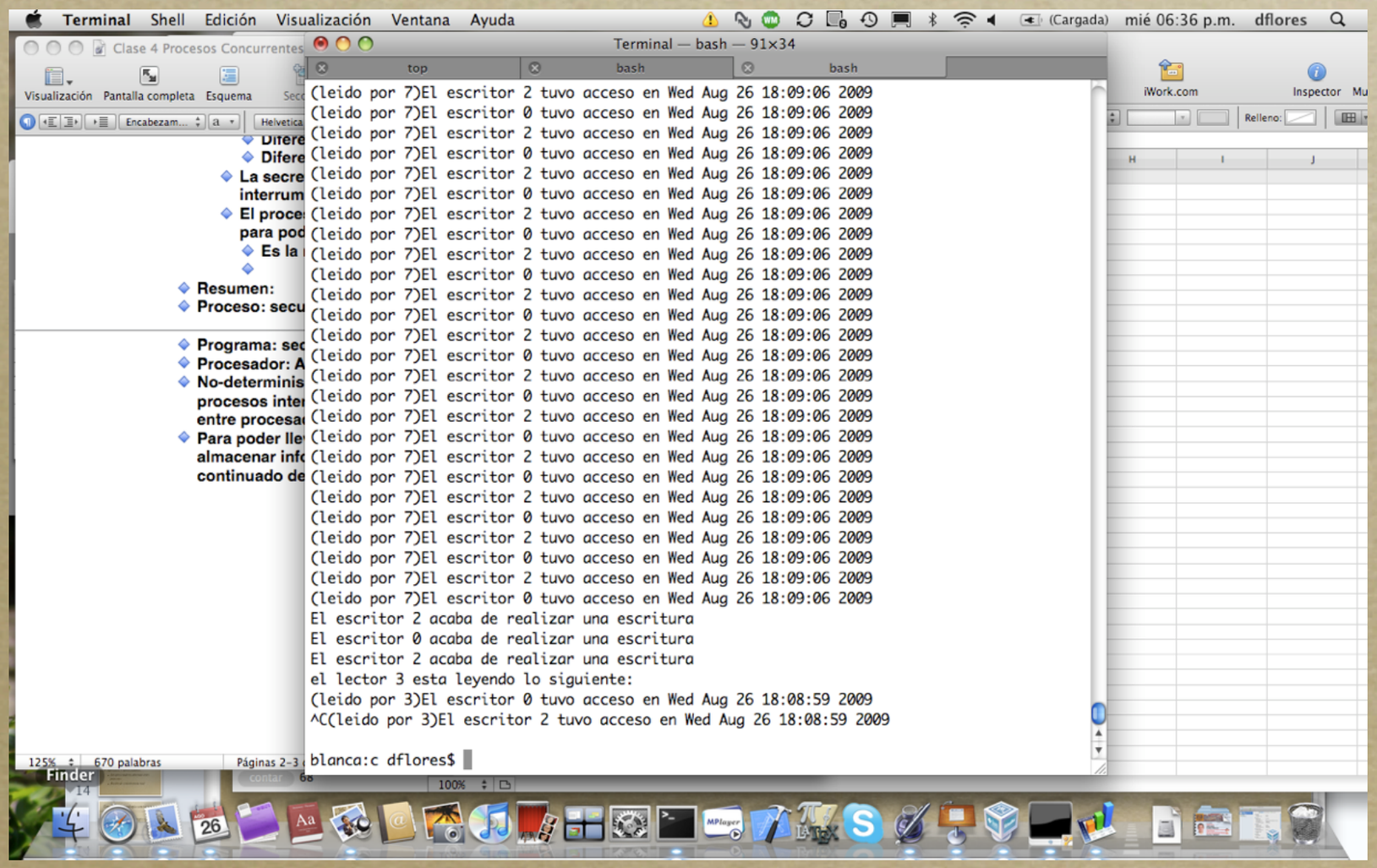
- *Una aplicación sofisticada consiste de varios procesos*
- *Hoja de cálculo incrustada aquí*

x	x^2
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100
11	121
12	144
13	169
14	196
15	225
16	256
17	289
18	324
19	361
20	400
21	441
22	484
23	529
24	576
25	625
26	676
27	729
28	784
29	841
30	900
31	961
32	1024
33	1089
34	1156

Concurrencia y no-determinismo

Concurrencia

- *Activación de varios procesos al mismo tiempo*



Concurrencia aparente

- *#procesos > #procesadores*
- *Los procesadores alternan entre procesos*
- *Ilusión de concurrencia real*

Concurrencia: analogía con secretario

- *Un solo secretario, múltiples tareas*
  - *Escribir documentos*
  - *Capturar dictado*
  - *Preparar café*
  - *Contestar teléfonos*

Concurrencia: analogía con secretaria

- *Un solo secretario, múltiples tareas*
  - *Solo hace una cosa en un momento dado*
  - *Salta de una tarea a otra*
    - *debe continuar la tarea interrumpida después*

Concurrencia

- *Si se toma una instantanea del sistema, hay más de un proceso entre su estado inicial y su estado final.*
  - *Incluye concurrencia real y aparente*

No-determinismo

- *Impredecibilidad de:*
  - *Instrucción en la que se interrumpe un proceso*
    - *Orden relativo de ejecución de instrucciones de procesos diferentes*
  - *Tiempo que tomará una operación de E/S*
    - *Orden/Tiempo de interrupciones*
  - *Solicitud de recursos de un proceso*
    - *Archivos/Sockets/Memoria/...*

No-determinismo/Secretario

- *Impredecibilidad de:*
  - *Llamado del jefe*
  - *Instante en que sonará un teléfono*
    - *Tiempo que tardará en contestar*

No-determinismo

- *Interrupción/intercambio de procesos*
  - *Debemos recordar el punto y el estado en el que está para poder regresar después a su ejecución*

Responsabilidades del SO respecto a procesos

- *Creación/destrucción*
- *Intercambio de procesos en los procesadores*
  - *De manera transparente!*

Sincronización de procesos

Semáforos

Los procesos...

- *Cooperan*
  - *Datos procesados por uno, son usados por otro*
- *Compiten*
  - *Memoria, disco, procesador...*

Los procesos...

- *Cooperan, Compiten por recursos*
- *Áreas en la que la sincronización/CIP es esencial:*
  - *Exclusión mutua*
  - *Sincronización*
  - *Prevención de Deadlocks*

Exclusión mutua

- *Hay recursos que no pueden usarse por dos procesos al mismo tiempo*
  - *Impresora*
  - *Localidades de memoria*

Agentes de reserva de boleto

- *Agente A observa si el asiento 5 está disponible, y pregunta a su cliente*
- *Agente B observa si el asiento 5 está disponible, y pregunta a su cliente*
- *Agente A reserva el asiento 5*
- *Agente B reserva el asiento 5*

Recursos…

- *… no compartibles*
  - *Mayoría de periféricos*
  - *Archivos escribibles*
    - *Recursos sujetos a modificación*
- *… compartibles*
  - *CPU's*
  - *Archivos de solo lectura*
    - *Recursos no modificables*

Exclusión mutua

- *El problema es asegurar que un solo proceso tiene acceso a cierto recurso en un momento dado*

Sincronización

- *La velocidad relativa de procesos es impredecible*
  - *Depende de la frecuencia con que son interrumpidos*
  - *=> Los procesos son <u>asíncronos</u>*
- *Algunos procesos se deben <u>sincronizar</u>*
  - *Productores/Consumidores*

Deadlock

- *Situación en que un grupo de procesos se bloquea*
- *Un proceso ocupa un recurso que el otro necesita para continuar*
- *Filosos cenando*

Problema de filósofos

- *Cada filósofo:*
  - *Toma un tenedor*
  - *Toma el otro tenedor*
  - *come*
  - *suelta tenedores*
  - *repite*

Comunicación interprocesos

- *Exclusión mutua*
- *Sincronización*
- *Deadlock*

Comunicación interprocesos

- *Exclusión mutua*
- *Sincronización*
- *Deadlock*

Semáforos!

Semáforos

- Edsger Dijkstra, 1965
- Un semáforo es una variable entera que se manipula con sólo dos operaciones *indivisibles.*
  - *wait(sem)*
  - *signal(sem)*

Semáforos

wait(sem):

while(sem==0);

decrementa sem;

signal(sem):

incrementa sem;

- *Tan pronto puede*, decrementa *el semáforo en* exactamente una unidad.

Incrementa *el semáforo*, *en* exactamente una unidad.

Indivisibilidad de wait(), signal()

sem := 3;

Semáforo		*Variable*	
Proceso A	*Proceso* B	*Proceso* A	*Proceso* B
Signal(sem);	*Signal(sem);*	*sem:=sem+1;*	*sem := sem +1;*

Indivisibilidad de wait(), signal()

sem := 3;

Semáforo

Proceso A
Signal(sem);

Proceso B
Signal(sem);

sem := 5;

Variable

Proceso A
sem:=sem+1;

Proceso B
sem := sem +1;

Indivisibilidad de wait(), signal()

sem := 3;

Semáforo

Proceso A
Signal(sem);

Proceso B
Signal(sem);

sem := 5;

Variable

Proceso A
sem:=sem+1;

Proceso B
sem := sem +1;

sem := 4;

wait(sem)

- *Retraso potencial (cuando sem == 0)*
  - *No hay retraso si sem >0.*
- *Potencial costo: cambio de contexto*

wait(sem)

- *Solo se puede continuar cuando algún signal() hace que sem != 0.*
  - *Un sólo proceso puede continuar*
    - *No sabemos cual!*

Invariante

- *cada signal()*, incrementa *el semáforo*.
- *cada wait() <u>completada</u>*, decrementa *el semáforo*
  - =>
    - *val(sem) == inicial(sem)+signals(sem) - waits(sem)*

Invariante

- *cada signal()*, incrementa *el semáforo*.
- *cada wait()* *completada*, decrementa *el semáforo*

- =>
  - *val(sem) == inicial(sem)+signals(sem) - waits(sem)*
- *val(sem) >= 0*
  - => *waits(sem) <= inicial(sem)+signals(sem)*

Invariante

- => $waits(sem) <= inicial(sem) + signals(sem)$

- *La igualdad se da si sem == 0.*

Semáforos

- *Indivisibilidad*
  - *-> Típicamente con soporte de HW.*

Exclusión mutua

- *El problema es asegurar que un solo proceso tiene acceso a cierto recurso en un momento dado*

Exclusión mutua

- *El problema es asegurar que un solo proceso tiene acceso a cierto recurso en un momento dado*
  - *Un semáforo por recurso!*

Exclusión mutua

- *El problema es asegurar que un solo proceso tiene acceso a cierto recurso en un momento dado*
  - *Un semáforo por recurso!*
    - *wait(sem_recurso);*
    - *uso de recurso*
    - *signal(sem_recurso);*
  - *Valor inicial del semáforo: 1.*

Sección crítica

- *La sección de código en la que el proceso usa el recurso no-compartible*
- *Analogía con un cuarto de baño*

Sección crítica

- *Cerca de metro C.U.*
  - *Un tramo de vía de dos sentidos*

Sección crítica

- *La sección de código en la que el proceso usa el recurso no-compartible.*
  - *waits(sem_recurso) <= signals(sem_securso) +1*

- *El número de procesos que han entrado, es a lo más el número de procesos que han salido, más uno.*

Consumidor/Productor manipulando una cola

- *Programa de proceso A*
- .
- .
- *wait(cola)*
- *añade caja a cola;*
- *signal(cola)*
- .
- .
-

- *Programa de proceso B*
- .
- .
- *wait(cola)*
- *quita caja de cola;*
- *signal(cola)*
- .
- .
-

Agentes de reserva de boleto

- *Agente A observa si el asiento 5 está disponible, y pregunta a su cliente*
- *Agente B observa si el asiento 5 está disponible, y pregunta a su cliente*
- *Agente A reserva el asiento 5*
- *Agente B reserva el asiento 5*

Agentes de reserva de boleto (2)

- *wait(asiento5)*
- *Agente A observa si el asiento 5 está disponible, y pregunta a su cliente*
- *Agente A reserva el asiento 5*
- *signal(asiento5)*

- *wait(asiento4)*
- *Agente B observa si el asiento 4 está disponible, y pregunta a su cliente*
- *Agente B reserva el asiento 4*
- *signal(asiento4)*

Agentes de reserva de boleto (2)

- *wait(autobus10)*
- *Agente A observa si el asiento 5 está disponible, y pregunta a su cliente*
- *Agente A reserva el asiento 5*
- *signal(autobus10)*

- *wait(autobus10)*
- *Agente B observa si el asiento 4 está disponible, y pregunta a su cliente*
- *Agente B reserva el asiento 4*
- *signal(autobus10)*

Agentes de reserva de boleto (2)

- *wait(terminal)*
- *Agente A observa si el asiento 5 está disponible, y pregunta a su cliente*
- *Agente A reserva el asiento 5*
- *signal(terminal)*

- *wait(terminal)*
- *Agente B observa si el asiento 4 está disponible, y pregunta a su cliente*
- *Agente B reserva el asiento 4*
- *signal(terminal)*