



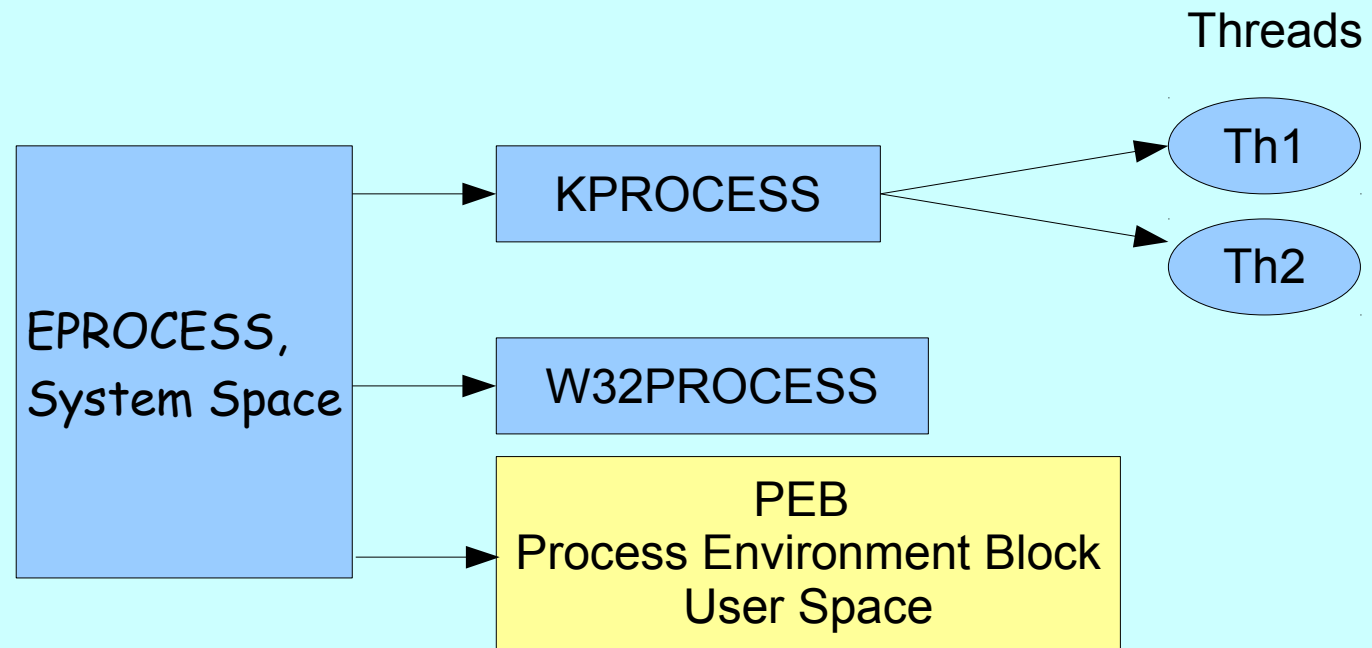
75-08 Sistemas Operativos
Lic. Ing. Osvaldo Clúa
2010

Facultad de Ingeniería
Universidad de Buenos Aires

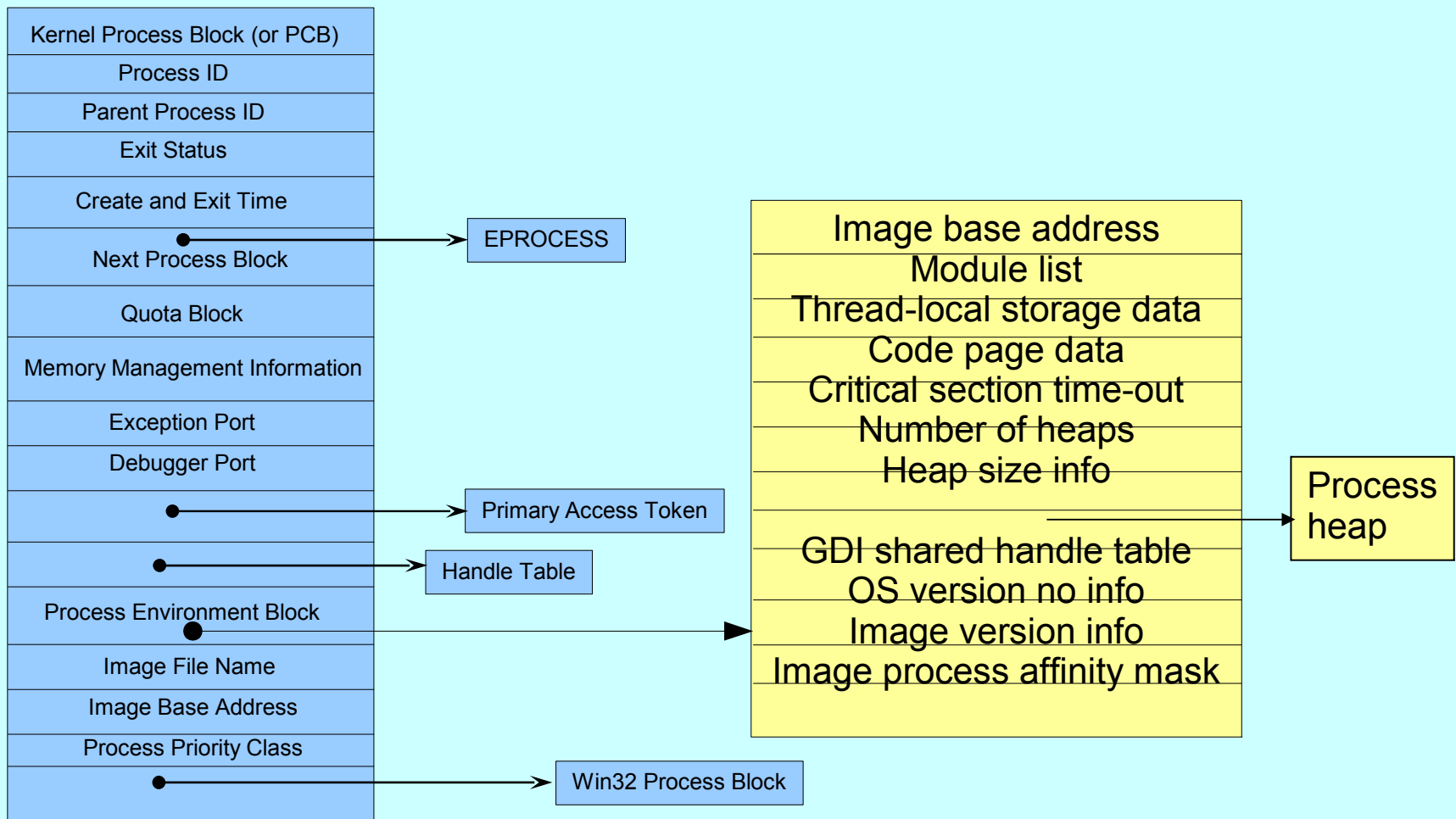
Procesos en Windows

EPROCESS

- Executive Process (bloque de control)
 - Es la representación de un proceso para Windows.
 - Son varias estructuras relacionadas.



Detalle del EPROCESS y del PEB



Flujo de CreateProcess

- Abrir el archivo imagen (.exe).
- Crear el proceso.
- Crear el thread inicial.
- Notificar al Windows Subsystem.
 - Termina la inicialización del thread (Carga de DLLs, etc).
- Comenzar la ejecución del programa.

Abrir el archivo imagen

- Definir la prioridad con la que se va a crear el proceso.
- Asociar el proceso a un Desktop.
- Verificar las políticas de seguridad (si el usuario tiene permitido correr esa imagen)
- Asociar el proceso con un subsistema (Win16, Win32, WoW, DOS)
 - Invocar la imagen correspondiente si no es la nativa.
 - Si hay un intérprete predefinido en el Registry (un debugger o un Just in Time Compiler), invocarlo.

Crear el Proceso

- Preparar el bloque EPROCESS.
- Preparar el espacio inicial de direcciones.
- Crear el bloque KPROCESS
- Mapea los recursos del executive al espacio de direcciones (Ntdll.dll, Nat. Language, etc).
- Prepara el PEB.
- Encadena el EPROCESS (aunque todavía no puede ejecutar).

Crear el Thread Inicial

- Incrementa la thread count del proceso.
- Prepara un Executive Thread Block (ETHREAD) y le asigna un id.
- Prepara un Thread Environment Block (TEB)
- Mapea la dirección de comienzo.
- Inicializa prioridades y características que dependen del sistema (afinidad, seguridad)

Notificar al Subsistema (Windows en este ejemplo)

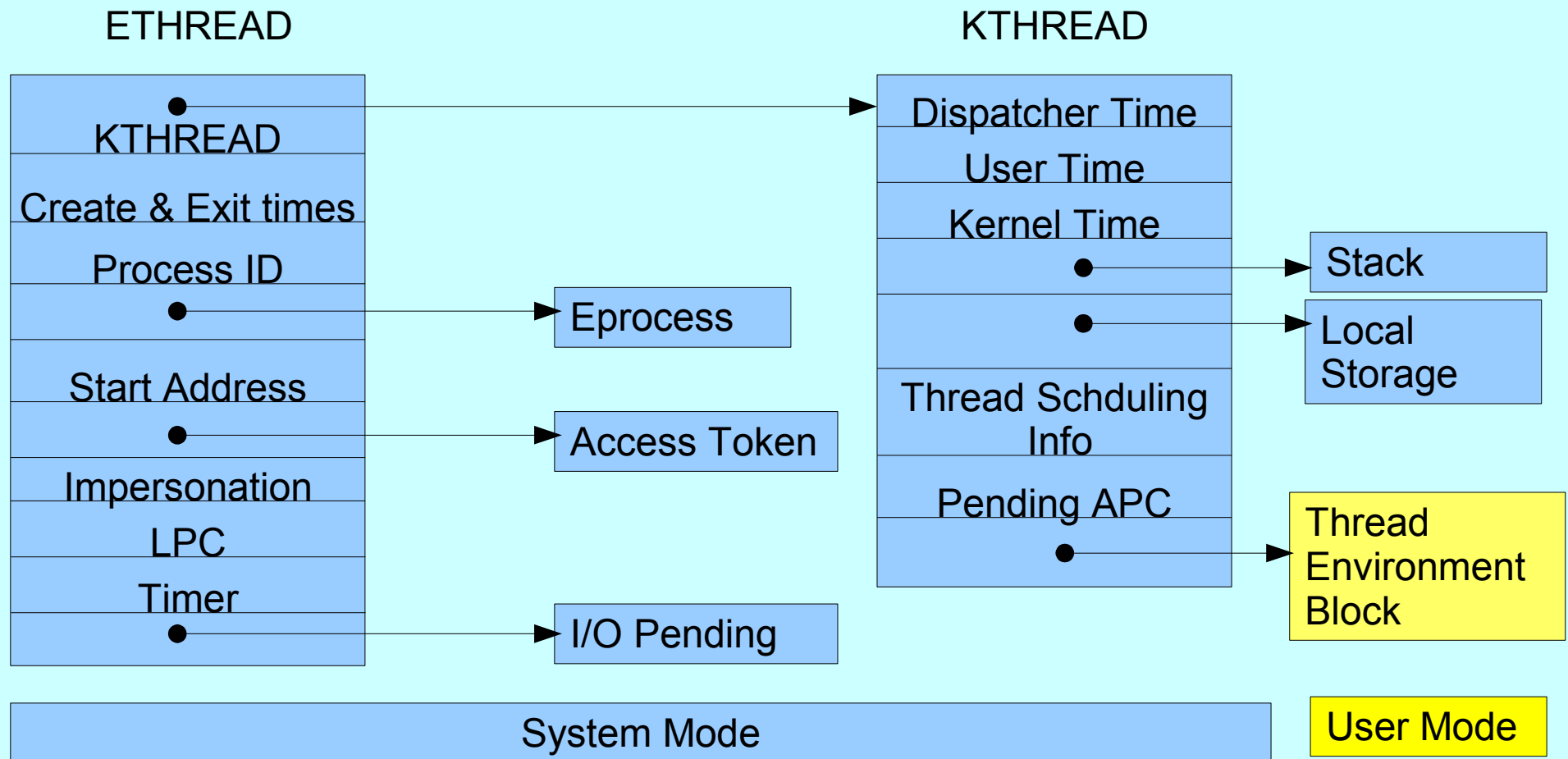
- Crea las estructuras para que CRSS lo maneje.
- Prepara el bloque W32PROCESS.
- Muestra el cursor de "starting" por 7 segundos.
 - Si el proceso no crea una ventana, vuelve al cursor original.



Pasos finales

- Inicializa el Heap y demás estructuras de run-time.
- Carga las DLLs necesarias ...
 - y las inicializa llamando a su entry-point `DLL_PROCESS_ATTACH`.
 - si es .net interpreta los XML asociados a las DLL (o assemblies).
- Prepara un Asynchronous Process Call (APC) para comenzar la ejecución.

ETHREAD



CreateThread

- Función Win32 en Kernel32.dll .
- Crea el stack e inicializa el contexto.
- Inicializa ETHREAD.
- Deja al thread en ready.
- Al resumir la ejecución
 - Ejecuta los "Pasos Finales" de la creación de un proceso.

Usando el Process Explorer para ver la actividad de un Thread (Totalcmd.exe)

The screenshot shows the Process Explorer window with the 'Process' list. The 'TOTALCMD.EXE' process is selected, showing a PID of 1236 and a CPU usage of 2.61. Below the process list, a table of loaded DLLs is visible, including advapi32.dll, apphelp.dll, atl.dll, browseui.dll, clbcatq.dll, comctl32.dll, comdlg32.dll, comres.dll, crypt32.dll, cryptui.dll, cscdll.dll, csui.dll, ctypes.dll, davclnt.dll, dprov.dll, gdi32.dll, ieframe.dll, iertutil.dll, imagehlp.dll, and imm32.dll.

The 'TOTALCMD.EXE:1236 Properties' dialog box is open, showing the 'Threads' tab. The thread list shows three threads:

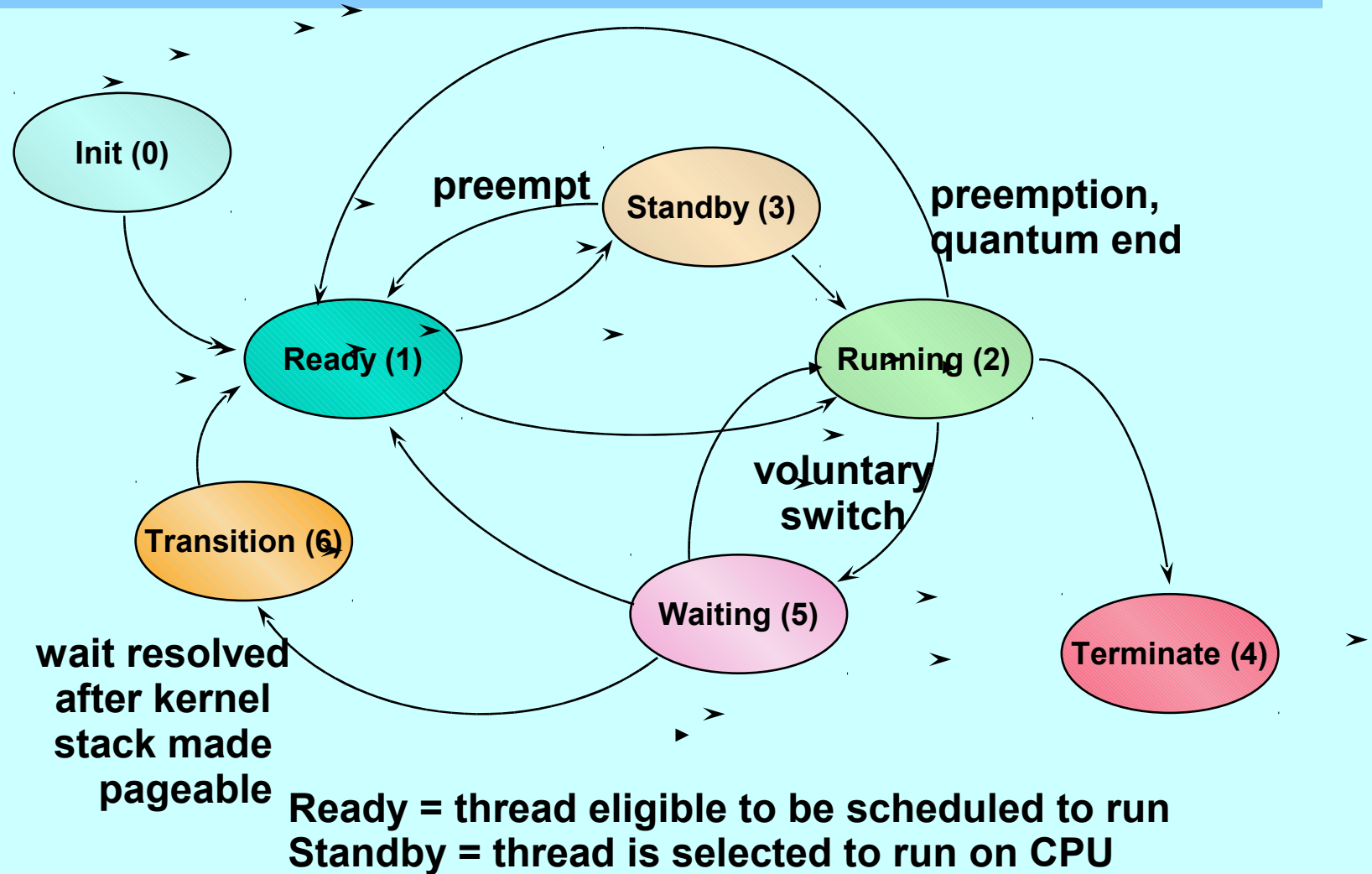
CPU	CSwitch Delta	Start Address
		TOTALCMD.EXE+0x31b500
		TOTALCMD.EXE+0x3624
		ntdll.dll!RtlQueueWorkItem+0x2b5

The 'Thread ID' is 1268. The 'Start Time' is 14:01:57 06/08/2008. The 'State' is 'Wait:UserRequest'. The 'Kernel Time' is 0:00:00.390. The 'User Time' is 0:00:00.030. The 'Context Switches' is 1.706. The 'Base Priority' is 6. The 'Dynamic Priority' is 8.

The 'Stack for thread 1268' dialog box is also open, showing the stack trace:

```
0 ntoskrnl.exe!ExReleaseResourceLite+0x206
1 ntoskrnl.exe!RtlAppendUnicodeToString+0x2b8
2 ntoskrnl.exe!CcSetDirtyPinnedData+0x3f0
3 ntoskrnl.exe!ProbeForWrite+0x505
4 ntoskrnl.exe!ZwYieldExecution+0xb78
5 ntdll.dll!KiIntSystemCall+0x6
6 USER32.dll!GetLastInputInfo+0x105
7 USER32.dll!MsgWaitForMultipleObjects+0x1f
8 TOTALCMD.EXE+0x1ad1cd
9 TOTALCMD.EXE+0x1aa218
10 TOTALCMD.EXE+0x364e
```

Estados de un Thread



Algunos estados del diagrama

- Standby - esperando por un determinado procesador
 - Solo un thread por procesador puede estar en standby.
- Waiting - a la espera de algún evento.
- Transition - Ready pero sus estructuras no están en memoria.

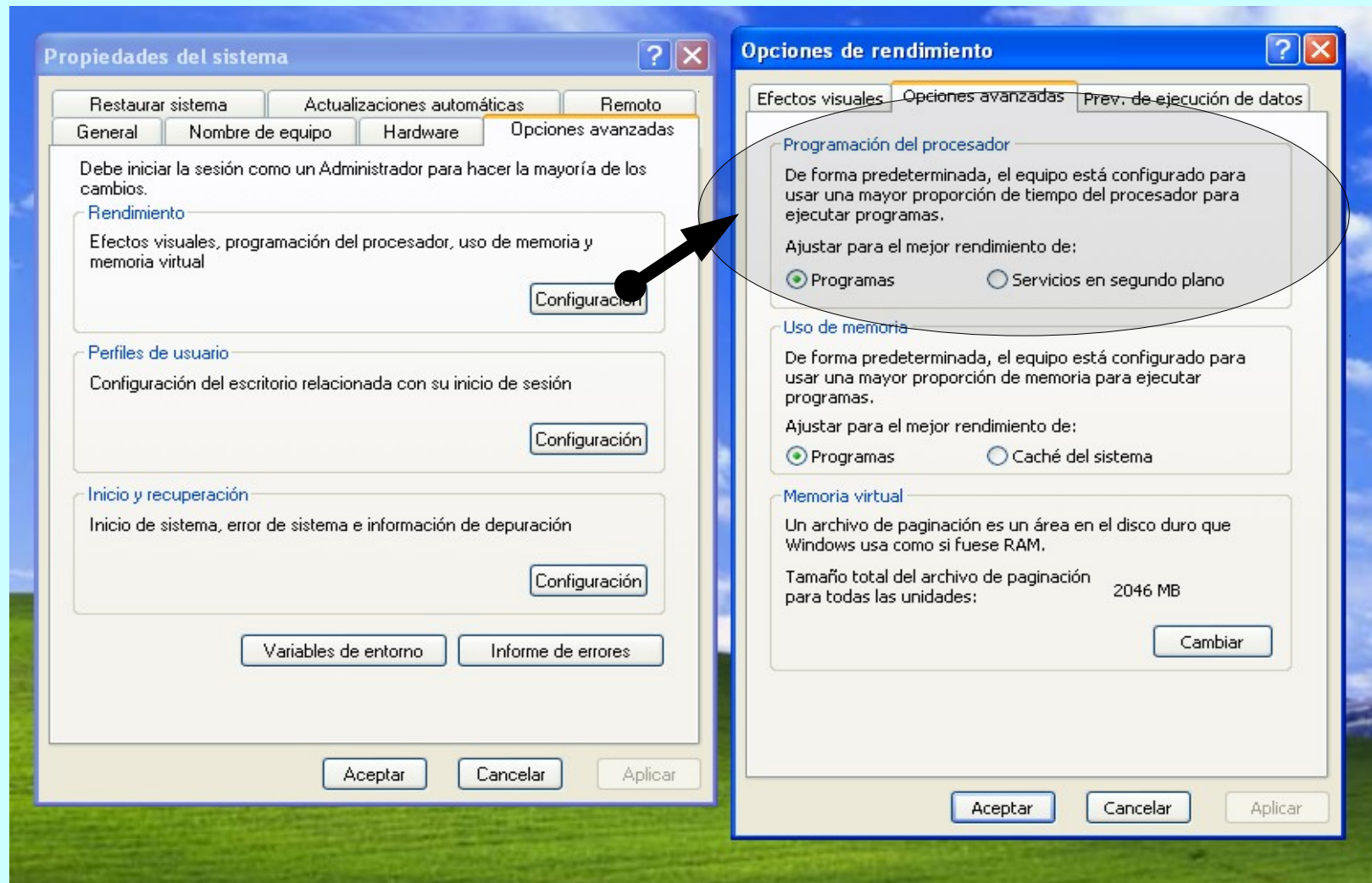
El Scheduler

- Está repartido por el Kernel.
 - Colectivamente se lo conoce como *Dispatcher*.
 - Utiliza Prioridades.
 - Los threads corren por un quantum de tiempo.
 - Es apropiativo (preemptive).
 - Trabaja a nivel de threads sin tener en cuenta al proceso

El Quantum

- Su valor es 3^* ticks de reloj
 - con cada tick se decrementa en 3 si el thread esta running y en 1 si esta waiting.
 - 6 para el Workstation, 36 para el Server.
 - Se puede ajustar entre ciertos límites.
 - Interactúa con la prioridad.

Eligiendo un Quantum



Idle Thread

- También conocido como Idle Process, System Process o Idle.
- Se lo activa cuando no hay threads listos para ejecutarse
 - Verifica y termina los DPC (Deferred Procedure Call) que hubiera.
 - Habilita interrupciones
 - Después de un tiempo llama a las rutinas de ahorro de energía.

Multiprocesadores

- Cualquier thread corre en cualquier CPU (a menos que se especifique *affinity*)
 - Se trata de mantenerlos en la misma CPU (soft affinity).
 - No hay un Master Processor.
- A partir de 2003 hay una cola ready por CPU y se tiene en cuenta **NUMA**.