Procesos y Planificación de la CPU

Proceso

Un **Proceso** puede informalmente entenderse como un programa en ejecución. Formalmente un proceso es _*Una unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual, y un conjunto de recursos del sistema asociados*_.

Para entender mejor lo que es un proceso y la diferencia entre un programa y un proceso, A. S. Tanenbaum propone la analogía _*Un científico computacional con mente culinaria hornea un pastel de cumpleaños para su hija; tiene la receta para un pastel de cumpleaños y una cocina bien equipada con todos los ingredientes necesarios, harina, huevo, azúcar, leche, etc.*_ Situando cada parte de la analogía se puede decir que la receta representa el programa (el algoritmo), el científico computacional es el procesador y los ingredientes son las entradas del programa. El proceso es la actividad que consiste en que el científico computacional vaya leyendo la receta, obteniendo los ingredientes y horneando el pastel.

*A lo largo de este resumen, serán sinonimos los conceptos de Job, Tarea y Proceso.*

Desambiguación

Un **Programa** es estático, no tiene program counter y existe desde que se edita hasta que se borra. Por otro lado, un **Proceso** es dinámico, tiene program counter y su ciclo de vida comprende desde que se _*dispara*_ hasta que termina.

Componentes de un Proceso

Proceso: Entidad de Abstracción

Un proceso (para poder ejecutarse) incluye como mínimo:
- Sección de Código (texto)
- Sección de Datos (variables globales)
- Stack(s) (datos temporarios: parámetros , variables temporales y direcciones de retorno)

Stacks

Un proceso cuenta con 1 o mas stacks

- En general, modo Usuario y modo Kernel
    - Se crean automáticamente y su medida se ajusta en run-time.
    - Está formado por stack frames que son pushed (al llamar a una rutina) y popped (cuando se retorna de ella)

    - El stack frame tiene los parámetros de la rutina(variables locales), y datos necesarios para recuperar el stack frame anterior (el contador de programa y el valor del stack pointer en el momento del llamado)

Atributos de un Proceso

- Identificación del proceso, y del proceso padre
- Identificación del usuario que lo *disparó*
- Si hay estructura de grupos, grupo que lo disparó
- En ambientes multiusuario, desde que terminal y quien lo ejecuto

Process Control Block (PCB)

Estructura de datos asociada al proceso (abstracción)

- Existe una por proceso.
- Es lo primero que se crea cuando se crea un proceso y lo último que se borra cuando termina
- Contiene la información asociada con cada proceso:
    - PID, PPID, etc
    - Valores de los registros de la CPU (PC, AC, etc)
    - Planificación (estado, prioridad, tiempo consumido, etc)
    - Ubicación (representación) en memoria
    - Accounting
    - Entrada salida (estado, pendientes, etc)

> Qué es el espacio de memoria de un Proceso?

- Es el conjunto de direcciones de memoria que ocupa el proceso
    - stack, text y datos
- No incluye su PCB o tablas asociadas
- Un proceso en modo usuario puede acceder sólo a su espacio de direcciones
- En modo kernel, se puede acceder a estructuras internas o a espacios de direcciones de otros procesos

El Contexto de un Proceso

Incluye toda la información que el SO necesita para administrar el proceso, y la CPU para ejecutarlo correctamente. Son parte del contexto, los registros
de cpu, inclusive el contador de programa, prioridad del proceso, si tiene E/S pendientes, etc

Context Switch

Se produce cuando la CPU cambia de un proceso a otro. Se debe resguardar el contexto del proceso saliente, que pasa a espera y retornará después

la CPU. Se debe cargar el contexto del nuevo proceso y comenzar desde la instrucción
siguiente a la última ejecutada en dicho contexto. Es tiempo no
productivo de CPU. El tiempo que consume depende del soporte de HW.

* * *

Sobre el Kernel del Sistema Operativo

Ver a partir de diapositiva 17

Estados de un Proceso

En su ciclo de vida, un proceso pasa por diferentes estados.

- Nuevo (new)
- Listo para Ejecutar (ready)
- Ejecutándose (running)
- En espera (waiting)
- Terminado (terminated)

Colas en la planificación de proceso

- Se enlazan las PCBs.

Módulos de Planificación

Los módulos de planificación son módulos de software del kernel del sistema operativo que realizan distintas tareas asociadas a la planificación.

Se ejecutan ante determinados eventos que así lo requieren:
- Creación/Terminación de procesos
- Eventos de Sincronización o de E/S
- Finalización de lapso de tiempo
- Etc

> Módulos de planificación

- **Scheduler de long term**

    - Controla el grado de multiprogramación, es decir, la cantidad de procesos en memoria.
    - Puede no existir este scheduler y absorber esta tarea el de short term.

- **Scheduler de short term**

    - Decide a cuál de los procesos en la cola de listos se elige para que use la CPU.
    - Términos asociados: apropiativo, no apropiativo, algoritmo de scheduling

- **Scheduler de medium term (swapping)**

    - Si es necesario, reduce el grado de multiprogramación.
    - Saca temporalmente de memoria los procesos que sea necesario para mantener el equilibrio del sistema.
    - Términos asociados: swap out (sacar de memoria), swap in (volver a memoria).

*Su nombre proviene de la frecuencia de ejecución.*

Otros módulos: **Dispatcher** y **Loader.**

- Pueden no existir como módulos separados de los schedulers vistos, pero la función debe cumplirse.
- **Dispatcher** hace cambio de contexto, cambio de modo de ejecución... _*despacha*_ el proceso elegido por el Short Term (es decir, _*salta*_ a la instrucción a ejecutar).
- **Loader** carga en memoria el proceso elegido por el long term.

Estados de Procesos

- **New**

Cuando se dispara un proceso: Un proceso es creado por otro proceso, su proceso padre. Se crean las estructuras asociadas quedando el proceso en la cola de procesos, normalmente en espera de ser cargado en memoria.

- **Ready**

El proceso queda en Estado Ready una vez que el Long Term Scheduler lo elige para cargarlo en memoria.

El proceso solo necesita que se le asigne la CPU.

Está en la Ready Queue.

- **Running**

El Short Term Scheduler le asigna la CPU. Tendrá la misma hasya que finalice el tiempo asignado (quantum o time slice), termine o hasta que necesite realizar una operación de E/S.

- **Waiting**

El proceso necesita que se cumpla el proceso esperado para continuar la ejecución. Sigue en memoria pero no tiene la CPU. Luego de esto pasará al estado de Ready.

> Transiciones entre Estados

- **New-Ready**: Por elección del scheduler de largo plazo (carga en memoria)
- **Ready-Running**: Por elección del scheduler de corto plazo (asignación de CPU)
- **Running-Waiting**: el proceso *_se pone a dormir_*, esperando por un evento.
- **Waiting-Ready**: Terminó la espera y compite nuevamente por la CPU

> Caso especial

- **Running-Ready**: Cuando el proceso termina su quantum (franja de tiempo) sin haber necesitado ser interrumpido por un evento, pasa al estado de ready, para competir por CPU, pues no está esperando por ningún evento...

*_Ver explicación de estados en diapositiva 22_*

Planificación

> Procesos alternan ráfagas de CPU y de I/O.

En estos alternados se dan los siguientes casos:

- CPU-bound
  - Mayor parte del tiempo utilizando la CPU
- I/O-bound (I/O = E/S)
  - Mayor parte del tiempo esperando por I/O
- La velocidad de la CPU es mucho mas rápida que la de los dispositivos de E/S
  - Pensar: Necesidad de atender rápidamente procesos I/O-bound para mantener el dispositivo ocupado y aprovechar la CPU para procesos CPU-bound

> Planificación

Surge la necesidad de determinar cual de todos los procesos que estan en estado de Ready será el próximo a ejecutarse en un ambiente multiprogramado. A partir de esto se plantean **algoritmos de planificación** del sistema.

Algoritmos de Planificación

Apropiativos

En los algoritmos Apropiativos (preemtive) existen situaciones que hacen que el proceso en ejecución sea expulsado de la CPU.

No Apropiativos

En los algoritmos No Apropiativo (nonpreemptive) los procesos se ejecutan hasta que el mismo (por su propia cuenta) abandone la CPU
- Se bloquea por E/S o finaliza
- No hay decisiones de planificación durante las interrupciones de reloj

> Según el ambiente es posible requerir algoritmos de planificación diferentes, con diferentes metas.

Equidad, Otorgar una parte justa de la CPU a cada proceso

Balance, Mantener ocupadas todas las partes del sistema

- Ejemplos:
    - Procesos por lotes (batch)
    - Procesos Interactivos
    - Procesos en Tiempo Real

Creación de Procesos

Un proceso es creado por otro proceso. Un proceso padre tiene uno o más hijos. Se forma un **arbol de procesos**.

Actividades de Procesos

- Crear la PCB
- Asignar PID único
- Asignarle memoria para regiones
    - Stack
    - Text
    - Datos

- Crear estructuras de datos asociadas
    - Se utiliza el Fork para copiar el contexto y asignar memoria para las regiones.

Relación entre procesos Padre e Hijo

> Con respecto a la ejecución

- El padre puede continuar ejecutándose concurrentemente con su hijo

- El padre puede esperar a que el proceso hijo (o los procesos hijos) terminen para continuar la ejecución.

> Con respecto al Espacio de Direcciones

- El hijo es un duplicado del proceso padre (caso Unix)
- Se crea el proceso y se le carga adentro el programa (caso Windows)

Creación de Procesos

En UNIX
- system call fork() crea nuevo proceso
- system call execve(), usada después del fork, carga un nuevo programa en el espacio de direcciones.

En Windows
- system call CreateProcess() crea un nuevo proceso y carga el programa para ejecución.

Terminación de procesos

Ante un (exit), se retorna el control al sistema operativo

- El proceso padre puede esperar recibir un código de retorno (via wait)
    - Proceso padre puede terminar la ejecución de sus hijos (kill)
- La tarea asignada al hijo se terminó
- Cuando el padre termina su ejecución
    - Habitualmente no se pemite a los hijos continuar, pero existe la opción.
    - Terminación en cascada

* * *

Procesos Cooperativos e Independientes

- **Independiente**: el proceso no afecta ni puede ser afectado por la ejecución de
otros procesos. No comparte ningún tipo de dato.
- **Cooperativo**: afecta o es afectado por la ejecución de otros procesos en el
sistema.

> Para que sirven los procesos cooperativos?

- Para compartir información (por ejemplo, un archivo)
- Para acelerar el cómputo (separar una tarea en sub-tareas que cooperan ejecutándose paralelamente)

- Para planificar tareas de manera tal que se puedan ejecutar en paralelo