***Definición de proceso:***

* Es un programa en ejecución.
* Sinónimos para nosotros: tarea, job, proceso.

***Diferencias entre un programa y un proceso:***

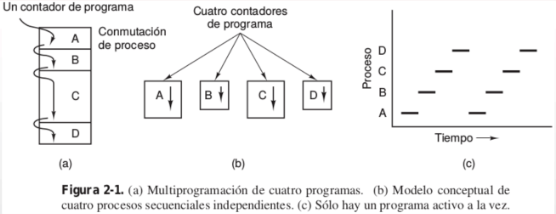
Programa:

* Estático.
* No tiene un program counter (posición de la instrucción que se ejecuta).
* Existe desde que se edita hasta que se borra.

Proceso:

* Dinámico.
* Tiene program counter.
* Su ciclo de vida comprende desde que se lo dispara hasta que termina.

***Modelo de proceso:***

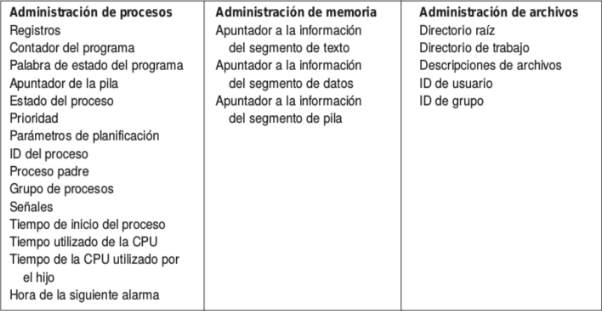
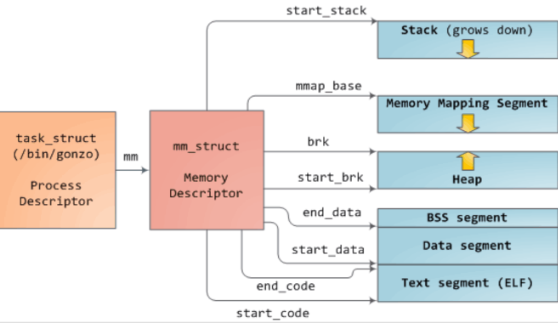


* Multiprogramación de 4 procesos.
* Modelo conceptual de 4 procesos secuenciales e independientes entre sí.
* Sólo un proceso está activo en cualquier instante (si tenemos una sola CPU).

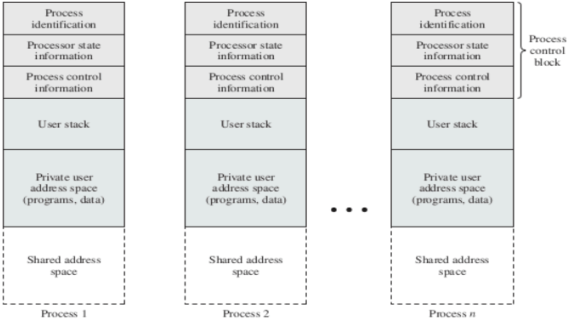
ANOTACION: El program counter es uno y los procesos tienen distintos valores en el.

***Componentes de un proceso:***

* Proceso: entidad de abstracción.
* Para poder ejecutarse dicho proceso, debe incluir como mínimo:
  + Sección de código (texto).
  + Sección de datos (variables globales)
  + Stack(s) (datos temporales: parámetros, variables temporales y direcciones de return).
    - Todo proceso cuenta con 1 o más stacks.
      * - En general: modo usuario (1 pila) y modo kernel (más de una pila) para dichos procesos.
    - Las pilas se crean automáticamente y su tamaño se ajusta en run-time.
    - Está formado por stack frames que son empujados (pushed- al llamar una rutina), y se retiran (popped- cuando se retorna de la rutina).
    - El stack frame tiene los parámetros de la rutina (variables locales), y datos necesarios para recuperar el stack frame anterior (PC y el valor del stack pointer en el momento del llamado).
* Atributos del proceso:
  + Identificador del proceso, y del proceso padre.
  + Identificador del usuario que lo “disparó”.
  + Si hay una estructura de grupos, grupo que lo disparó.
  + En ambientes multiusuario, desde que terminal y quién lo ejecutó.
* Process Control Block (PCB):
  + Es una estructura de datos asociada al proceso.
  + Existe una PCB por proceso.
  + Tiene referencias a memoria.
  + Es lo primero que se crea cuando se crea un proceso y lo último que se borra cuando termina.
  + El PCB lo podemos pensar como un gran registro en el que se guardan los atributos anteriormente mencionados, también guarda punteros y direcciones de memoría relacionadas al proceso.
  + Información asociada a cada proceso:
    - PID, PPID, etc.
    - Valores de los registros de la CPU (PC, AC, etc).
    - Planificación (estado, prioridad y tiempo consumido del proceso, etc).
    - Ubicación (donde está el proceso) en memoria.
    - Accounting (cantidades, cuanta memoria ocupó, cuanta entrada salida ocupó).
    - Entrada / salida (estado, pendientes, etc).



***¿Qué es el espacio de direcciones de un proceso?***

* Conjunto de direcciones de memoría que ocupa dicho el proceso.
  + Stack, código, datos.
* No incluye su PCB o tablas asociadas.
* Un proceso en modo usuario puede acceder SÓLO a su espacio de direcciones (limitado).
* En modo kernel, se puede acceder a estructuras internas (PCB del proceso por ej) o a espacios de direcciones de otros procesos (rompe el límite).
* 

***Contexto del proceso:***

* Incluye toda la información (esta en la PCB) que el SO necesita mantener para administrar el proceso, y en consecuente, el CPU pueda ejecutarlo correctamente.
* Son parte del contexto: los registros de CPU, inclusive el contador de programa (PC), prioridades del proceso, si tiene E/S pendientes, etc.
* Cambio de contexto (Context Switch):
  + Se produce cuando la CPU cambia de un proceso a otro.
  + Se DEBE resguardar el contexto del proceso anterior, que pasa a esperar para volver a ejecutarse después en la CPU. En PCB, otros quedan en el Stack.
  + 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 (la CPU debe estar optimizada para no sufrir tanto consumo de ciclos).
  + El tiempo que consume dicho cambio depende del soporte del Hardware.

1. Resguardo contexto del que se a.
2. Cargo contexto del entrante.
3. Paso a modo usuario
4. Retomo ejecución desde la instrucción siguiente.

***Datos sobre el kernel del S.O:***

* Conjunto de modulos de software.
* Se ejecuta en el procesador como cualquier otro proceso (NO ES UN PROCESO).
* Tiene varios enfoques de diseño:
  + Kernel como entidad independiente:
    - Kernel que se ejecuta fuera de todo proceso.
    - Arquitectura (diseño) usada por los primeros S.O.
    - Cuando un proceso es interrumpido o realiza una System Call, el contexto del proceso se salva y el control se pasa al kernel del sistema.
    - Genera una carga de tiempo no productivo en el CPU.
    - Es un modelo sencillo.
    - Tiene su propia región de memoría.
    - Tiene su propio stack.
    - Una vez finalizada su actividad, devuelve el control al proceso (o a otro proceso diferente).
    - Como importante:
      * El kernel si bien se parece a un proceso, NO es un proceso.
      * Se ejecuta como entidad independiente en modo privilegiado.
  + Kernel “dentro” de todos los procesos:
    - El código del kernel se encuentra dentro del espacio de direcciones de cada proceso.
    - El kernel se ejecuta en el MISMO contexto que algún proceso de usuario (no genera cambios de contexto, más rápido y eficiente).
    - El kernel se puede ver como una colección de rutinas que el proceso utiliza.
    - A no confundirse, no se repite en muchos espacios de memoría, sino que los procesos conocen donde se encuentra el kernel.
    - Usa una pila de usuario, y otra para el modo kernel.
    - Dentro de un proceso se encuentra el código del programa (usuario), y el código de los módulos del kernel (privilegio)
    - El código del kernel se comparte por todos los procesos (como una libreria).
    - Cada interrupción (incluyendo llamadas al sistema), son atendidas en el contexto del proceso que se encuentra en ejecución.
      * Constante cambio entre modo usuario y modo kernel.
* Desventaja:
  + Le limito el espacio al proceso. De las 2^32 direcciones las parto a la mitad. Si un proceso ocupa más, se aclara en el compilador.