Introducción a los Sistemas Operativos

Procesos - I

Profesores:
Lía Molinari
Juan Pablo Pérez
Nicolás del Rio

Facultad de Informática
UNIVERSIDAD NACIONAL DE LA PLATA

- ☑ Versión: Agosto 2019
- ☑ Palabras Claves: Procesos, PCB, Stack, Contexto, Espacio de Direcciones

Los temas vistos en estas diapositivas han sido mayormente extraídos del libro de Andrew S. Tanenbaum (Sistemas Operativos Modernos) y del libro de William Stallings (Sistemas Operativos: Aspectos internos y principios de diseño)

Definición de proceso

- ☑ Es un programa en ejecución
- ☑ Para nosotros serán sinónimos: tarea, job y proceso

Diferencias entre un programa y un proceso

Programa

- Es estático
- No tiene *program counter*
- Existe desde que se edita hasta que se borra

Proceso

- Es dinámico
- Tiene *program counter*
- Su ciclo de vida comprende desde que se solicita ejecutar hasta que termina

El Módelo de Proceso

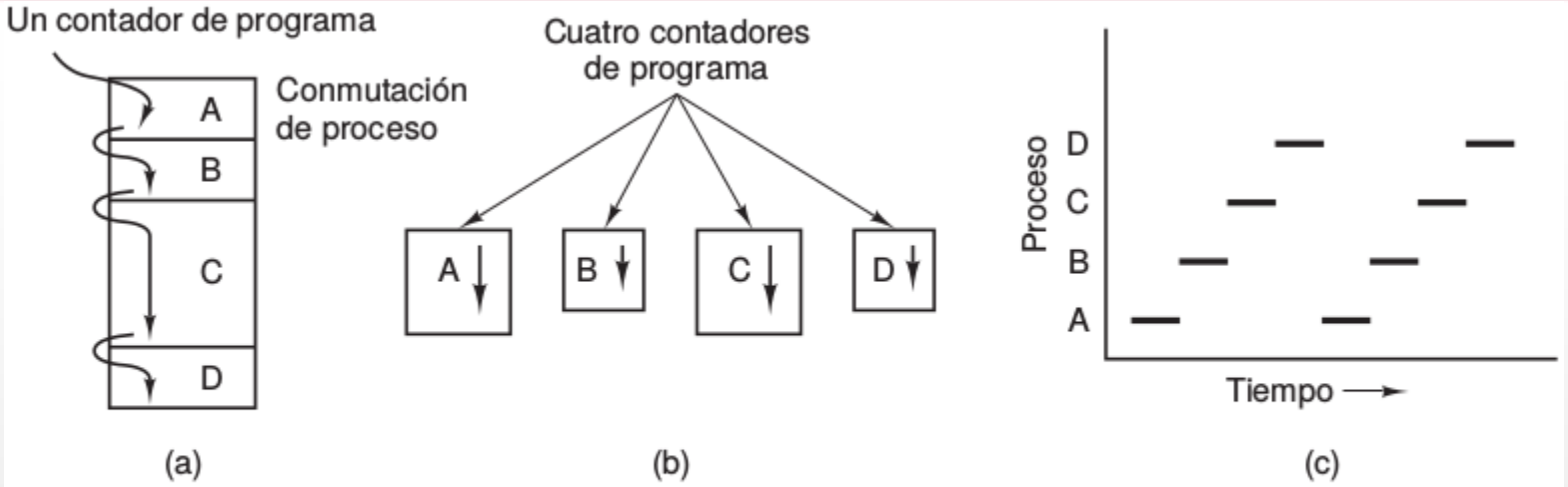


Figura 2-1. (a) Multiprogramación de cuatro programas. (b) Modelo conceptual de cuatro procesos secuenciales independientes. (c) Sólo hay un programa activo a la vez.

- ☑ Multiprogramación de 4 procesos
- ☑ Modelo conceptual de 4 procesos secuenciales e independientes.
- ☑ Solo un proceso se encontrara activo en cualquier instante. (Si tenemos una sola CPU)

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)

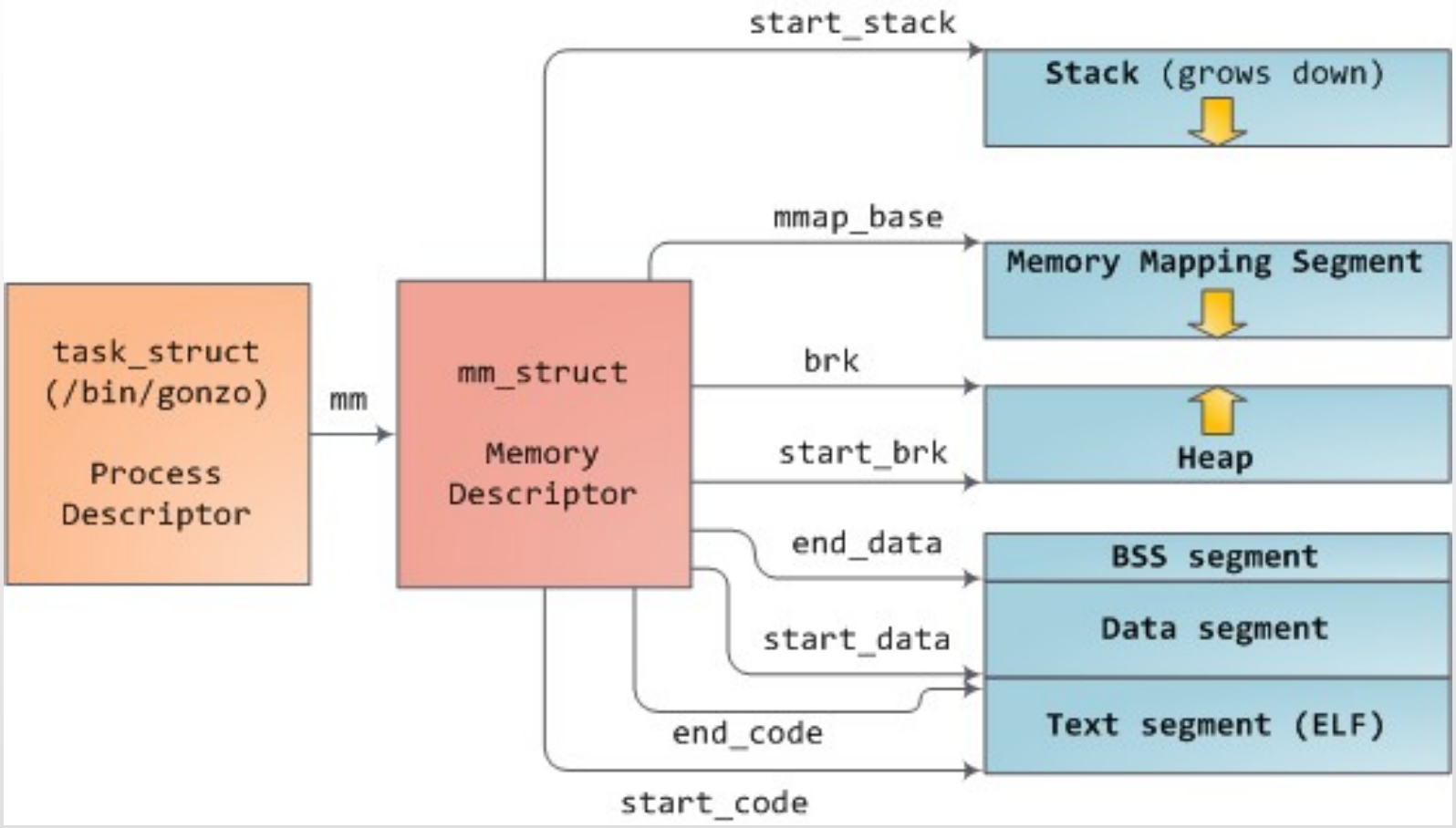
- Estructura de datos asociada al proceso (abstracción)
- Existe una por proceso.
- Es lo primero que se crea cuando se crea un proceso y lo último que se borra cuando termina
- Contiene la información asociada con cada proceso:
  - PID, PPID, etc
  - Valores de los registros de la CPU (PC, AC, etc)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

PCB (cont.)

Administración de procesos	Administración de memoria	Administración de archivos
Registros	Apuntador a la información del segmento de texto	Directorio raíz
Contador del programa	Apuntador a la información del segmento de datos	Directorio de trabajo
Palabra de estado del programa	Apuntador a la información del segmento de pila	Descripciones de archivos
Apuntador de la pila		ID de usuario
Estado del proceso		ID de grupo
Prioridad		
Parámetros de planificación		
ID del proceso		
Proceso padre		
Grupo de procesos		
Señales		
Tiempo de inicio del proceso		
Tiempo utilizado de la CPU		
Tiempo de la CPU utilizado por el hijo		
Hora de la siguiente alarma		

Campos Comunes

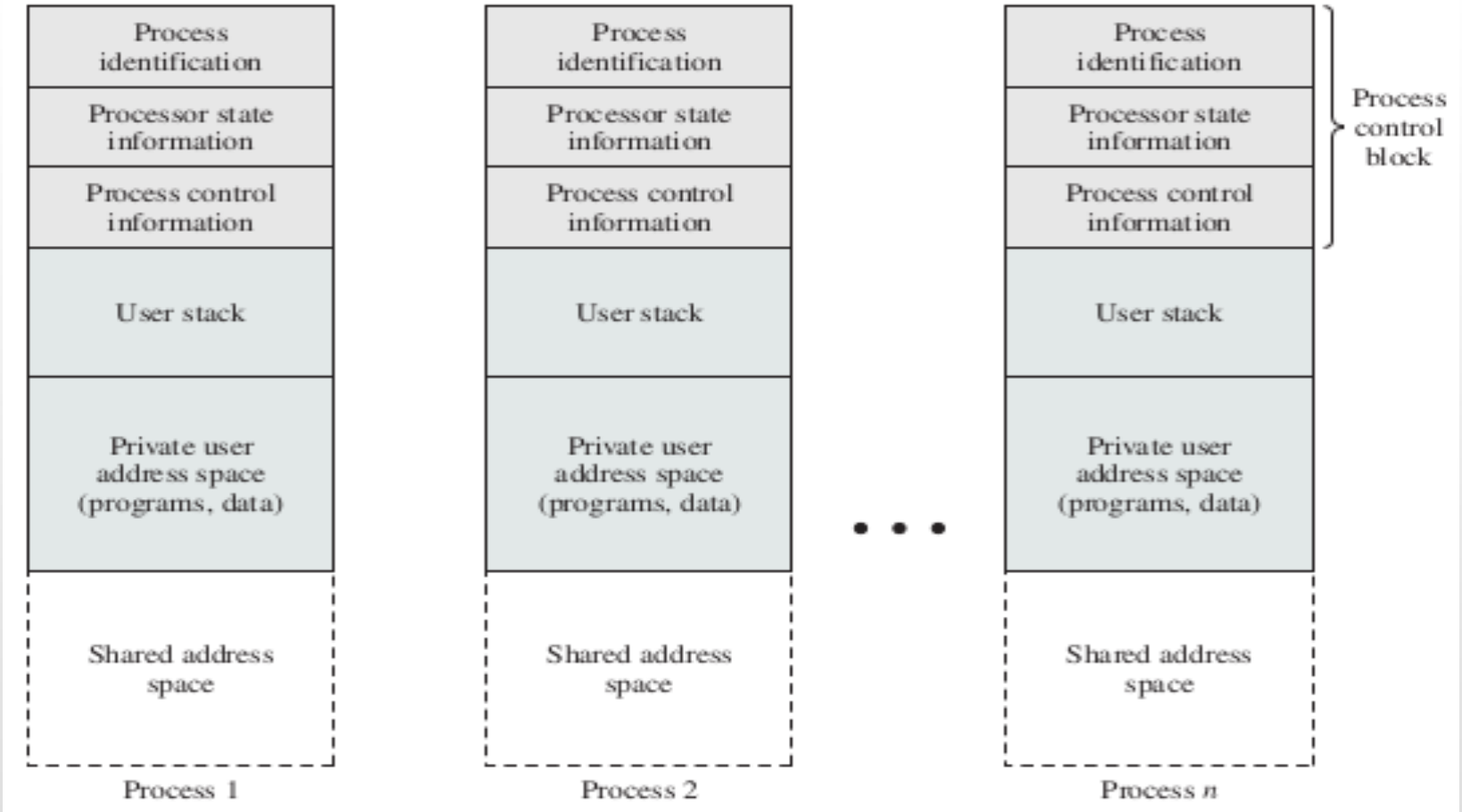
PCB (cont.)


start_stack
Stack (grows down)
mmap_base
Memory Mapping Segment
task_struct
(/bin/gonzo)
Process
Descriptor
mm
mm_struct
Memory
Descriptor
brk
start_brk
Heap
end_data
BSS segment
start_data
Data segment
end_code
Text segment (ELF)
start_code

Qué es el espacio de direcciones 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 (PCB del proceso por ejemplo) o a espacios de direcciones de otros procesos.

Espacio de direcciones del proceso + PCB



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,

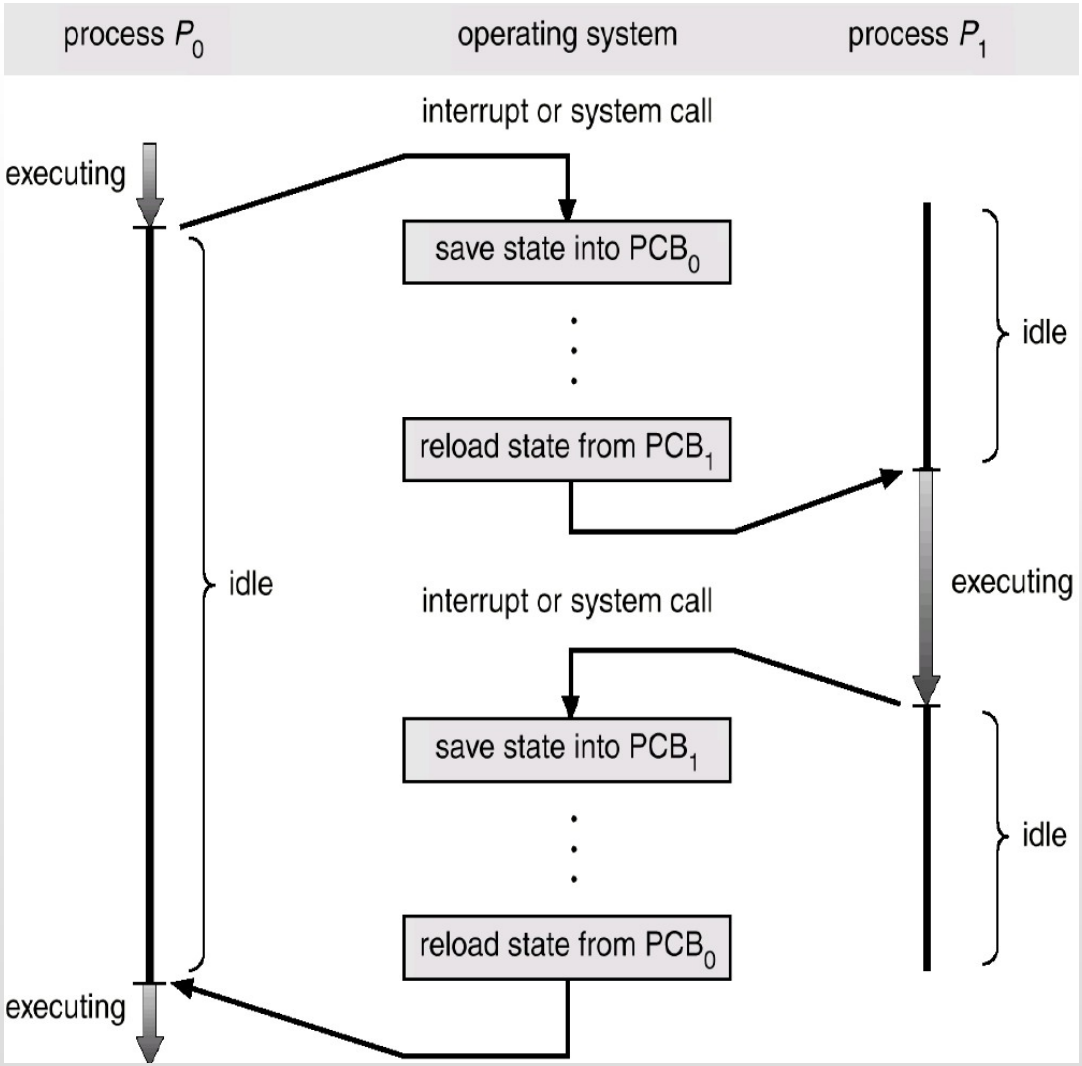
etc.

Administración de procesos	Administración de memoria	Administración de archivos
Registros	Apuntador a la información del segmento de texto	Directorio raíz
Contador del programa	Apuntador a la información del segmento de datos	Directorio de trabajo
Palabra de estado del programa	Apuntador a la información del segmento de pila	Descripciones de archivos
Apuntador de la pila		ID de usuario
Estado del proceso		ID de grupo
Prioridad		
Parámetros de planificación		
ID del proceso		
Proceso padre		
Grupo de procesos		
Señales		
Tiempo de inicio del proceso		
Tiempo utilizado de la CPU		
Tiempo de la CPU utilizado por el hijo		
Hora de la siguiente alarma		

Cambio de Contexto (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 a 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

Ejemplo de Cambio de Contexto

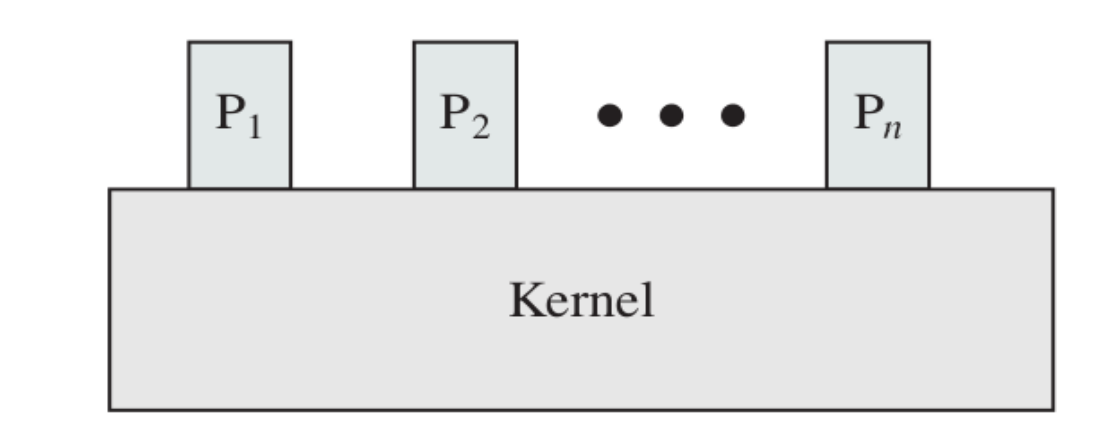


Sobre el Kernel del Sistema Opetativo

- ☑ Es un conjunto de módulos de software
- ☑ Se ejecuta en el procesador como cualquier otro proceso
- ☑ Entonces:
  - ¿El kernel es un proceso? Y de ser así ¿Quien lo controla?
- ☑ Diferentes enfoques de diseño

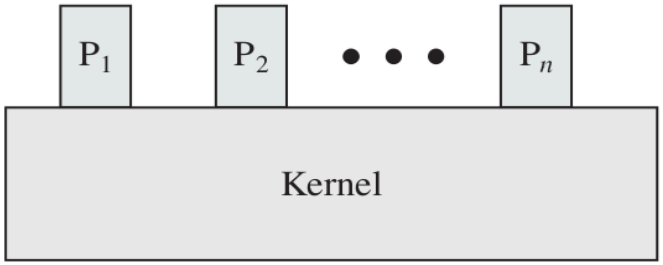
Enfoque 1 – El Kernel como entidad independiente

- El Kernel se ejecuta fuera de todo proceso
- Es una arquitectura utilizada por los primeros SO
- 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 operativo.



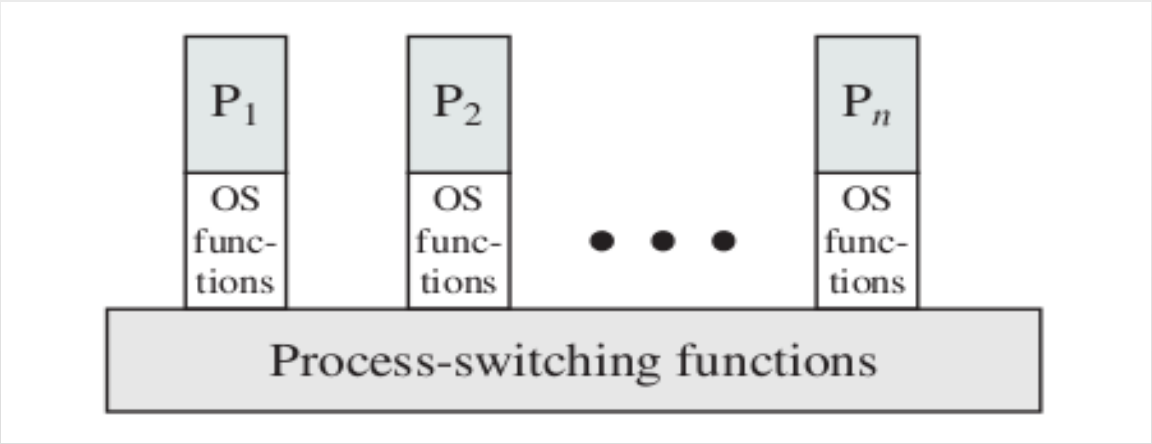
Enfoque 1 – El Kernel como entidad independiente

- El Kernel tiene su propia región de memoria
- El Kernel tiene su propio Stack
- Finalizada su actividad, le devuelve el control al proceso (o a otro diferente)
- Importante:
  - El Kernel NO es un proceso. EL concepto de proceso solo se asocia a programas de usuario
  - Se ejecuta como una entidad independiente en modo privilegiado



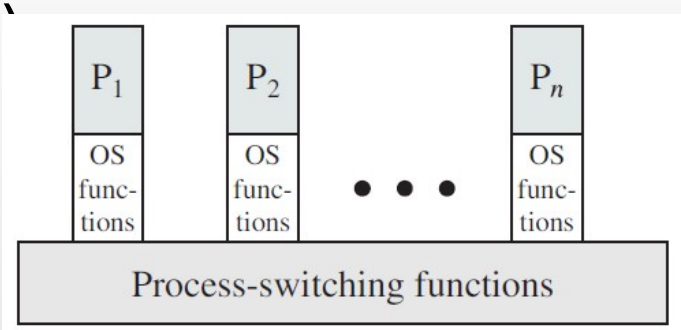
Enfoque 2 – El Kernel "dentro" del Proceso

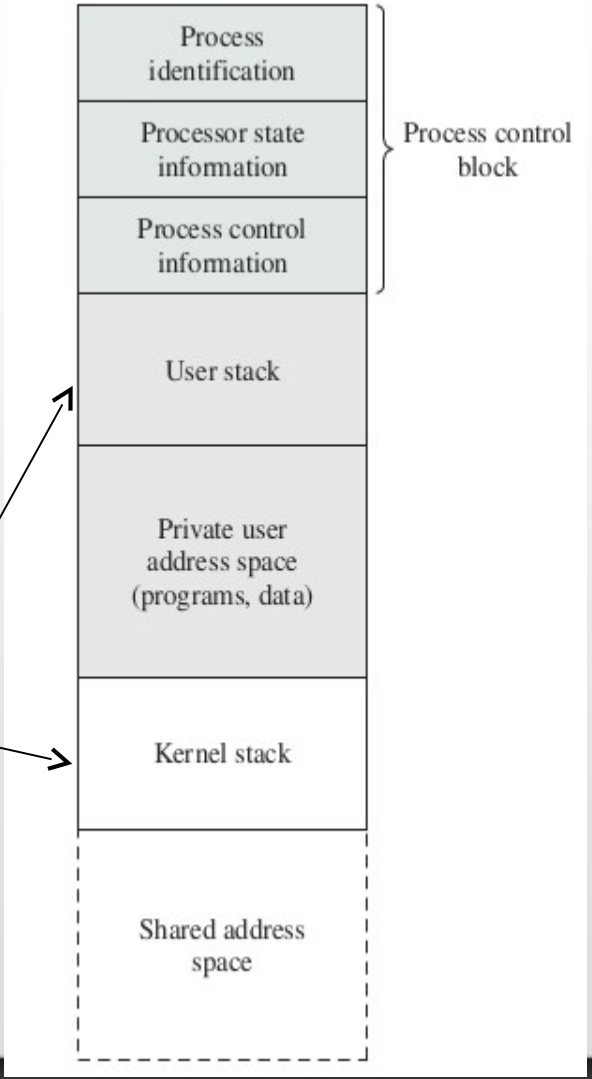
- El "Código" del Kernel se encuentra dentro del espacio de direcciones de cada proceso.
- El Kernel se ejecuta en el MISMO contexto que algún proceso de usuario
- El Kernel se puede ver como una colección de rutinas que el proceso utiliza



Enfoque 2 – El Kernel “dentro” del Proceso

- Dentro de un proceso se encuentra el código del programa (user) y el código de los módulos de SW del SO (kernel`





- Cada proceso tiene su propio stack (uno en modo usuario y otro en modo kernel)
- El proceso es el que se Ejecuta en Modo Usuario y el kernel del SO se ejecuta en Modo Kernel **(cambio de modo)**

Enfoque 2 – El Kernel "dentro" del Proceso

- El código del Kernel es compartido por todos los procesos
  - En administración de memoria veremos el "como"
- Cada interrupción (incluyendo las de System Call) es atendida en el contexto del proceso que se encontraba en ejecución
  - Pero en modo Kernel!!! (se pasa a este modo sin necesidad de hacer un cambio de contexto completo)
  - Si el SO determina que el proceso debe seguir ejecutándose luego de atender la interrupción, cambia a modo usuario y devuelve el control. Es mas económico y performante