# Sistemas Operativos

## Procesos

### Descripción y Control

Prof. Dr. Wenceslao Palma M.<wenceslao.palma@ucv.cl>

Gran parte de las acciones de un Sistema Operativo giran en torno a los procesos.

El sistema operativo permite la ejecución concurrente (intercalada) de muchos procesos en beneficio de un tiempo de respuesta razonable y maximizando la utilización del procesador.

El sistema operativo debe asignar recursos a los procesos de acuerdo a una política y evitando situaciones no deseables como el deadlock.

El sistema operativo ayuda en la estructuración de aplicaciones de usuario en cuestiones relacionadas con la creación de procesos y la comunicaión entre procesos.

Una clasificación según como los procesos comparten la memoria es:

Procesos Pesados: no comparten memoria. Cada uno se ejecuta en su procesador virtual. La ventaja es que esto proporciona protección. La comunicación entre procesos se realiza mediante pipes, mensajes, disco, etc.

Procesos Livianos: comparten memoria, no proporcionan protección y se comunican de manera eficiente mediante la memoria.

Otra clasificación tiene que ver con quién tiene el control para transferir el procesdor de un proceso a otro:

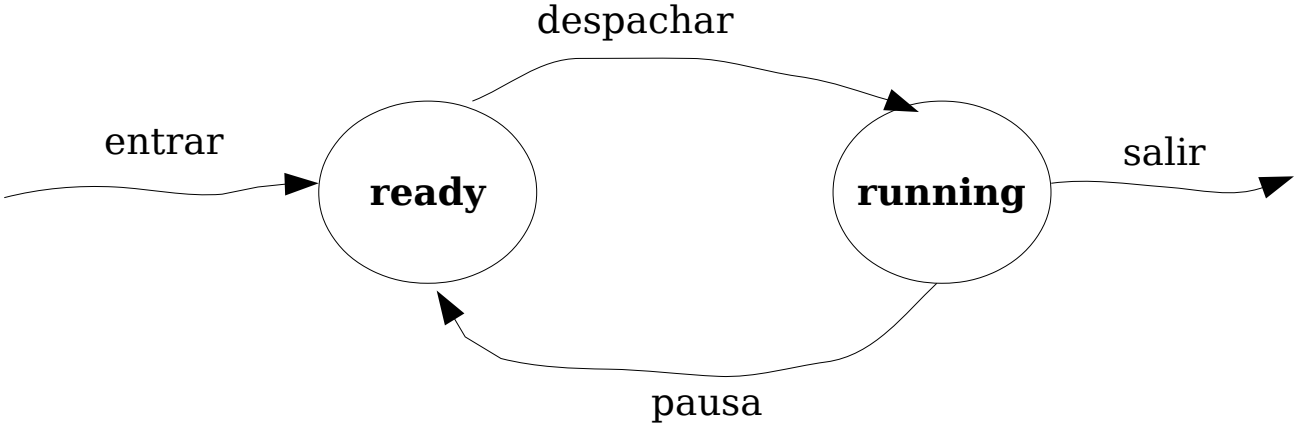
Procesos preemptive: el kernel toma la decisión de cual es el proceso que utilizará el procesador. Lo anterior puede ocurrir en cualquier momento.

Procesos non-preemptive: es el proceso quién toma la decisión de retornar el control del procesador a otro proceso.

## Estados de un proceso

Para poder controlar (asignación de cpu y otros recursos) los procesos es necesario caracterizarlos de acuerdo a su comportamiento.

Una manera sencilla de mirar esto es considerar que un proceso tiene dos estados **en ejecución (running)** y **no ejecución (ready)**.

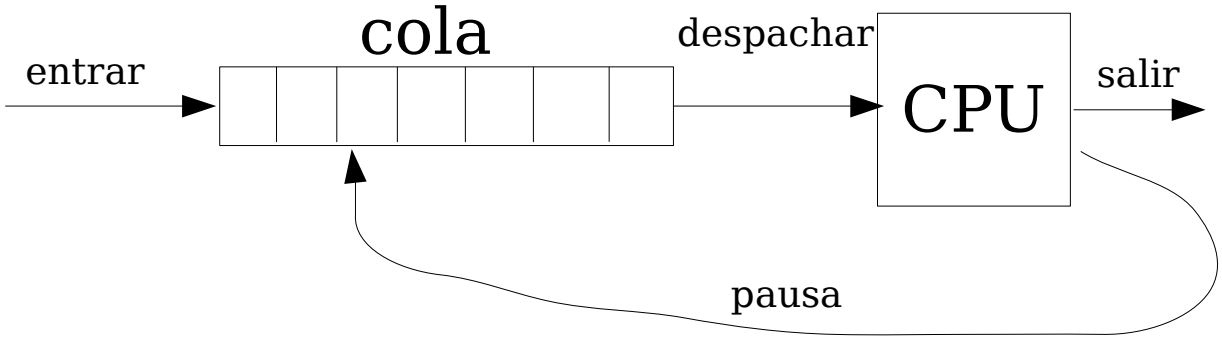


En el estado ready, el proceso existe y está esperando la oportunidad de ejecutarse.

El proceso que está en ejecución será interrumpido y el sistema operativo seleccionará a un proceso en estado ready para su ejecución.

El proceso interrumpido pasa desde el estado running a ready, y el proceso entrante de ready a running.

Lo anterior sugiere que el sistema operativo maneje información reativa a los procesos como su estado actual y posición en la memoria. Además, la selección del proceso que pasa del estado ready al estado running se debe fundamentar en una política clara.



## Creación y término de los procesos

Cuando se crea un nuevo proceso, se deben instanciar las estructuras de datos necesarias para su administracón y asignarle un espacio de direcciones en la memoria.

Es posible que un proceso puede originar la creación de otro proceso. Por ejemplo: el proceso servidor de impresión puede crear un proceso para atender a las solicitudes de impresión.

Cuando el sistema operativo crea un proceso tras la solicitud explícita de otro, la acción se conoce como **process spawning**

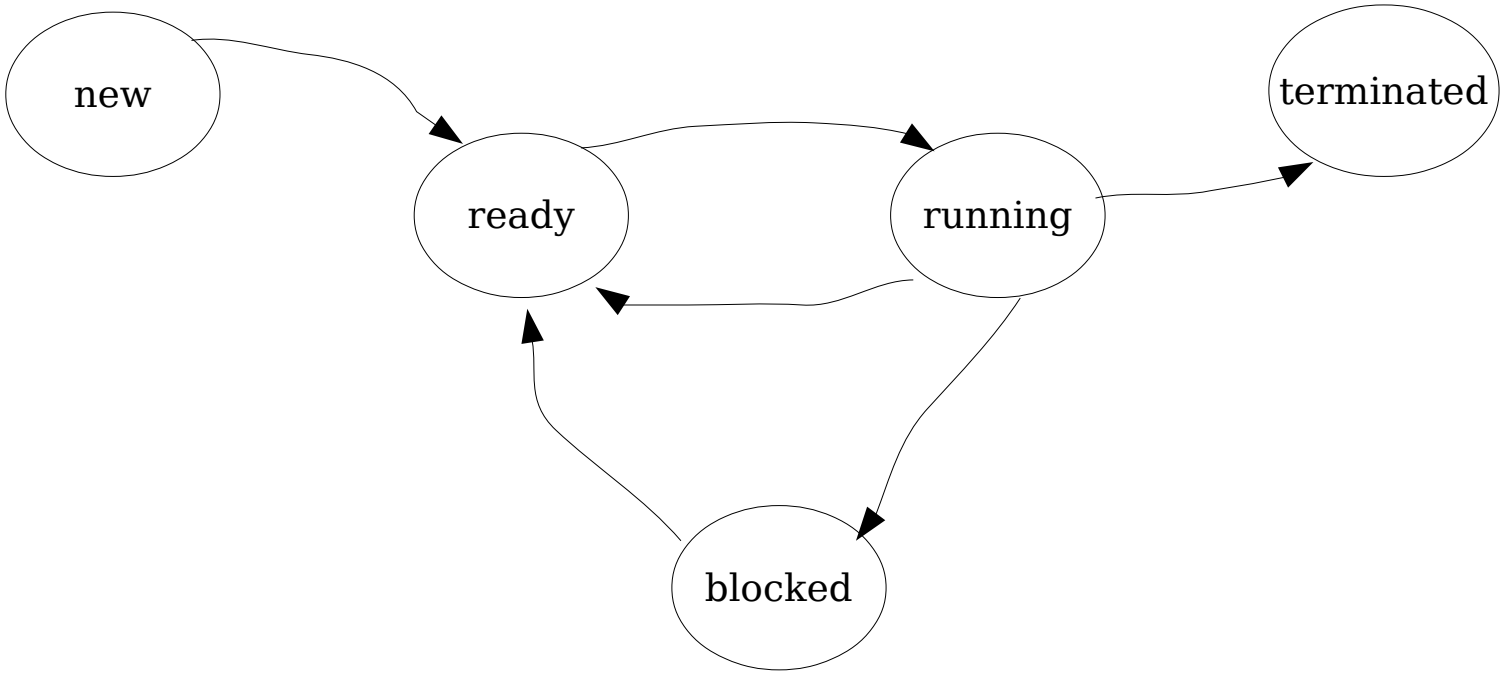
El proceso que genera a otro se conoce como **padre** y el generado como **hijo**.

Las razones para que un proceso termine son variadas:

- Termino normal.
- Tiempo límite excedido.
- No hay memoria disponible.
- Violación de límites.
- Tiempo máximo de espera sobrepasado.
- Error de E/S.
- Instrucción ilegal.
- Instrucción privilegiada.
- Mal uso de datos.
- Término/Solicitud del padre.

Mientras un proceso se ejecuta puede pasar por distintos estados, el modelo de dos estados presentado anteriormente no es suficiente.

Por lo anterior el modelo más aceptado y utilizado es el siguiente:




new: el kernel está obteniendo los recursos que necesita el proceso para poder ejecutarse.
ready: el proceso está preparado para ejecutarse en cuanto tenga la oportunidad.
running: el proceso está en ejecución, es decir, ejecutando sus instrucciones.
blocked: el proceso espera por el término de una operación de E/S, que llegue un mensaje de otro proceso, que termine otro proceso, etc.
terminado: el proceso no es elejible para su ejecución, pero existe para que otros procesos puedan determinar  que terminó. Luego, se borra de la memoria.

## Descripción de procesos

Esencialmente el sistema operativo administra el uso que hacen los procesos de los recursos del sistema.

Qué necesita el sistema operativo para controlar los procesos y administrar los recursos para ellos?

### Tablas de control del sistema operativo

El sistema operativo construye y mantiene tablas con los datos de cada entidad que administra.

Básicamente los datos se organizan en cuatro categorías: memoria, E/S, archivos y procesos.

Tabla de Memoria: se utilizan básicamente para seguir la pista de la memoria principal y secundaria asignada a cada proceso.

Tabla de E/S: utilizada para administrar los dispositivos y canales relacionados con E/S. Antes de acceder a un dispositivo es necesario conocer su estado, si hay una operación de E/S en marcha el sistema operativo necesita saber el estado de la operación y las direcciones de memoria utilizadas para la transferencia.

Tabla de archivos: almacenan datos sobre la posición en memoria principal y secundaria, estado actual y atributos.

Tabla de procesos: contiene datos necesarios para la administración de procesos.

Las tablas antes mencionadas deben mantener alguna relación ya que memoria, dispositivos y archivos son administrados en función de los requerimientos de los procesos.

El sistema operativo mediante asistencia del administrador o un programa de autoconfiguración tiene acceso a datos que definan su entorno básico.

**Estructuras necesarias para el control de los procesos**

Para administrar los procesos el sistema operativo debe mantener una estructura de datos que proporcione información relativa a ID, estado, ubicación en memoria, etc.

Asociado a un proceso se tiene: código, constantes, variables globales y locales, un stack para manejar invocación a funciones y un bloque de control de proceso (process control block, PCB).

Todo lo anterior se conoce como la imagen de un proceso.

La ubicación de la imagen de un proceso en la memoria, en su caso más simple, se almacena en un bloque de manera contigua. El bloque se mantiene en memoria secundaria.

Para que un proceso sea administrado, una parte de su imagen debe estar en memoria principal. Y para ser ejecutado debe estar completamente en memoria principal. Con esto es necesario que el sistema operativo conozca la ubicación tanto en memoria principal como secundaria de los procesos.

## PCB

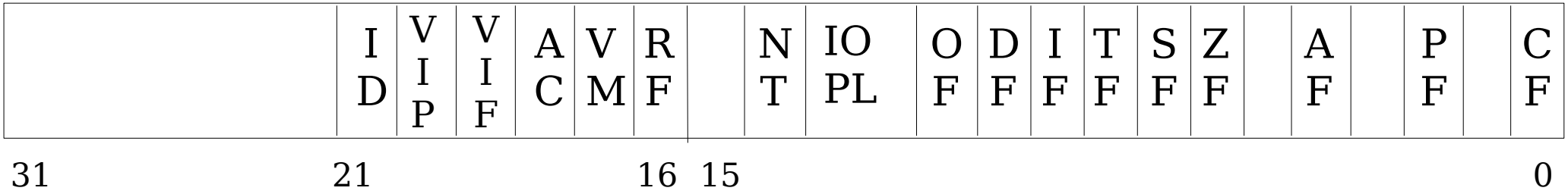
Es mantenida por el núcleo. El detalle de los datos allí almacenados varía con cada implementación de sistema operativo.

En general la información que allí se almacena se puede clasificar en: identificación de proceso, estado del procesador y control de proceso.

Identificación del proceso : cada proceso posee un número (PID) que lo identifica de manera única. También se almacena el identificar del proceso padre (PPID) y del usuario (UID). El PID se puede utilizar para mantener las referencias cruzadas entre las tablas almacenadas por el sistema operativo.

Información del estado del procesador: esta informaciómm se encuentra en los registros del procesador. Mientras un proceso está en ejecución, su información está en los registros, si este se detiene los registros deben salvarse para poder restaurar el proceso cuando se reanude su ejecución.

La información de estado se almacena en lo que se denomina PSW (Program Status Word). En máquinas Pentium se conoce como registro EFLAGS




**Bits de control**

- AC
- ID
- RF
- IOPL
- DF
- IF
- TF

## Bits de modo de operación

- NT
- VM
- VIP
- VIF

## Códigos de condición

- AF
- CF
- OF
- PF
- SF
- ZF

Información de control del proceso : almacena información relativa al estado del proceso, prioridad, scheduling, privilegios, memoria, etc.

**Contenido típico de un PCB**

Identificación de proceso :
- PID
- PPID
- UID

Información de estado del procesador :
- registros visibles para el usuario
- registros de control y estado
- punteros de pila

Información de control del proceso:
- scheduling y estado
- estructuración de datos
- comunicación entre procesos
- privilegios
- gestión de memoria
- propiedad de recursos y utilización

# Control de procesos

## Modos de ejecución

Se crean debido a la necesidad de proteger las estructuras de datos del sistema operativo ante los programas de usuario.

Existen dos modos: usuario y kernel (sistema, control)

En modo kernel se tiene el control completo del procesador, instrucciones, registros y memoria.

Hay un bit en la PSW que indica el modo. Este bit cambia ante distintos sucesos, por ejemplo ante la llamada al sistema por parte de un programa de usuario.

## Creación de procesos

La creación de procesos puede considerar los siguientes pasos:

1.- Asignar PID: significa agregar una entrada a la tabla de procesos.
2.- Asignar espacio al proceso: programas+datos+stack+PCB+enlaces para espacios de direcciones compartidos.
3.- Setear PCB: PID, PPID, contador de programa, punteros de stack, estado del proceso (ready), prioridad, etc.

4.- Enlaces: básicamente los relacionados con scheduling.
5.- Crear/Ampliar estructuras de datos: con el objeto de registrar información de accounting y/o evaluación de rendimiento.

## Cambio de contexto

Esta acción la realiza el scheduler para transferir el procesador de un proceso a otro.

Un cambio de contexto se origina en los siguientes sucesos: interrupción, Cepo, llamada al sistema.

Interrupción: el control se transfiere a un gestor de interrupciones y luego se salta a la rutina del sistema operativo que se ocupa del tipo de interrupción provocada. Algunas interrupciones:

de reloj: el sistema operativo determina si el proceso en ejecución ha consumido su quantum de tiempo. Si es así el proceso pasa al estado ready y se itinera otro proceso.

de E/S: si la acción involucra varios estados que estan bloqueados, estos pasan al estado ready. El sistema operativo decide si reanuda la ejecución del proceso actualmente en ejecución o da paso a otro (en estado ready) de mayor prioridad.

fallo de memoria: ocurre cuando el procesador descubre que una referencia a una dirección de memoria virtual no se encuentra en memoria principal. El sistema operativo trae el bloque a memoria principal, lo cual involucra una solicitud de E/S. Es posible que se provoque un cambio de contexto, luego que el bloque se almacena en memoria el proceso pasa al estado ready.

Cepo: tiene que ver con una condición de error o excepción generada dentro del proceso en ejecución. El sistema operativo determina si el error es fatal. Si es así, el proceso pasa al estado terminado y se produce un cambio de contexto.

Llamada al sistema: provoca la transferencia a una rutina que es parte del kernel, lo cual provoca, en la mayoría de los casos, que el programa usuario pase al estado blocked.

**Cambio de modo**

Cuando hay una interrupción pendiente, ocurre lo sigiuente:

1.- Salvar el contexto del programa en ejecución.
2.- Asigna al contador de programa el valor de la dirección de inicio de la rutina de la interrupción.
3.- Cambiar a modo kernel para que el código de la rutina de interrupción pueda incluir instrucciones privilegiadas.

Por lo anterior debe salvarse la parte del PCB relacionada con la información del estado del procesador, lo que incluye contador de programa, otros registros del procesador y el puntero al stack.

## Cambio de estado en los procesos

Cuando un proceso que se encuentra en ejecución (running) tiene que cambiar de estado (ready, blocked), el sistema operativo debe realizar cambios en su entorno:

1.- Salvar el contexto del procesador, incluye PC y otros registros.
2.- Actualizar el estado del proceso en PCB.
3.- Mover el PCB a la cola apropiada.
4.- Seleccionar otro proceso para su ejecución.
5.- Actualizar PCB del proceso seleccionado. Ahora el estado es running.
6.- Actualizar las estructuras de datos relacionadas con la gestión de la memoria.
7.- Restaurar el contexto del procesador al que existía en el momento que el proceso abandonó su estado running.

## Procesos en Unix versión V (SVR4)

Unix emplea tiene dos categorías de procesos: de sistema y de usuario.

Los procesos del sistema se ejecutan en modo kernel para realizar tareas como intercambio de procesos y reserva de memoria.

Los procesos de usuario se ejecutan en modo usuario para ejecutar programas y utilidades. Cuando un proceso de usuario realiza una llamada al sistema, se genera una interrupción o fallo  pasa a modo núcleo.

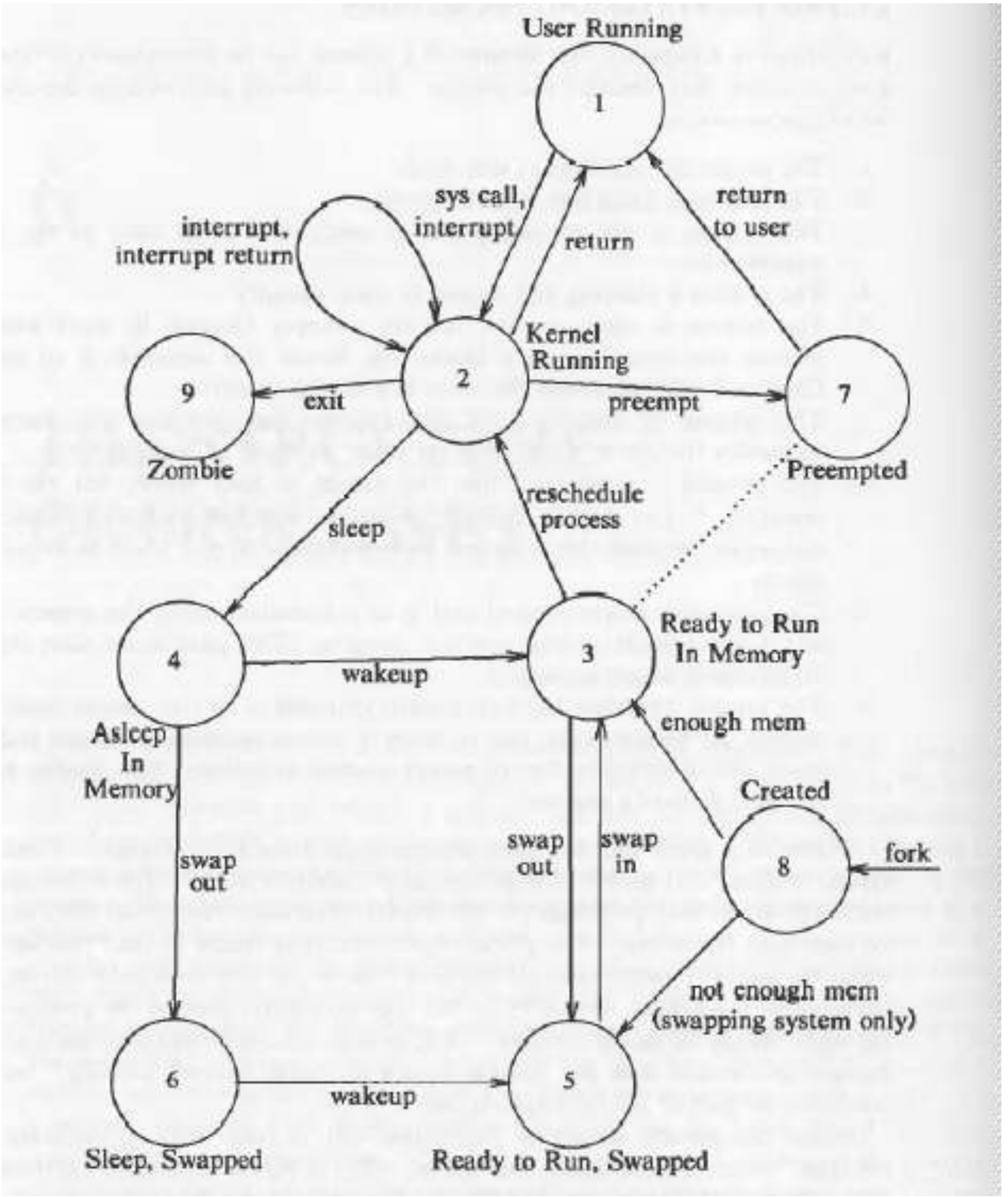
Hay dos procesos únicos en Unix: el proceso 0 y el proceso 1.

El proceso 0 se denomina proceso de intercambio (swapper), se crea cuando el sistema bootea. Además, crea el proceso 1 conocido como **init**, todos los procesos del sistema tienen a init como padre.

Unix gestiona los procesos en base a un diagrama de estados compuesto de 9 estados:

1.- el proceso se ejecuta en modo usuario.
2.- el proceso se ejecuta en modo kernel.
3.- el proceso no está en ejecución pero está preparado para correr cuando el kernel lo decida.
4.- El proceso se encuentra durmiendo en la memoria principal.

5.- El proceso se encuentra listo para ejecutar pero el swapper debe traerlo a memoria principal.
6.- El proceso está dormido y fue enviado por el swapper a memoria secundaria.
7.- El proceso retorna de modo kernel a modo usuario pero el kernel transfiere el procesador a otro proceso provocando un cambio de contexto.
8.- El proceso es creado y se encuentra en una fase de transición, no está dormido pero tampoco listo para ejecutarse. Esto se cumple para todos los procesos excepto el proceso 0.
9.- El proceso ejecutó la llamada al sistema exit y queda en estado zombie. El proceso ya no existe pero deja un registro que contiene su código de salida y otra información para su padre.




**Transición de estados en Unix**

Un proceso ingresa al sistema en el estado creado cuando su padre ejecuta la llamada al sistema **fork**. Eventualmente pasa al estado 3 ó 5.

Si se encuentra en el estado 3 el kernel, mediante el itinerador de procesos, lo selecciona y pasa a ejecutarse en modo kernel, donde completa la llamada a fork. Cuando esta llamada se completa pasa a ejecutarse en modo usuario.

Luego de un tiempo el reloj puede interrumpir el procesador y el proceso cambia a modo kernel. Cuando la interrupción termina el kernel puede seleccionar otro proceso para ejecutar, ante lo cual se provoca un cambio de contexto y el proceso saliente pasa al estado preempted.

Cuando el proceso es seleccionado por el kernel abandona el estado preempted y retorna a modo usuario.

Estando en modo kernel un proceso puede pasar al estado sleep mientras espera por un recurso (operación de I/O) o un fallo de página. luego de esto es despertado y pasa al estado listo para ejecutarse.

Eventualmente, debido a falta de memoria, un proceso que se encuentra durmiendo puede ser enviadoa disco (sleep swapped) y una vez que es despertado queda en estado listo para ejecutarse pero en memoria secundaria.

## Descripción de procesos

En Unix, la descripción de un proceso (imagen) se organiza en tres partes: contexto del usuario, contexto de los registros y contexto del sistema.

Contexto del usuario

Se puede generar a partir del código compilado. La zona de código es de solo lectura. Mientras el proceso se ejcuta, el procesador utiliza la zona de pila para las llamadas funciones, retorno y paso de parámetros. Cuando un proceso no está en ejecución, la información de estado del procesador se almacena en a zona de contexto de los registros. Los datos del contexto de usuario son:

código del proceso: sus intrucciones de máquina.

datos del proceso: var locales, globales.

pila de usuario: argumentos, var locales y puteros de las funciones que se ejecutan en modo usuario.

memoria compartida: para la comunicación entre procesos, es administrada gracias a la memoria virtual.

Contexto de los registros

contador de programa: puede estar en la memoria del kernel o del usuario.

registro de estado del procesador. contiene el estado del hardware al momento del cambio de contexto.

puntero de pila

registros de propósito general

Contexto del sistema

Contiene el resto de información necesaria para administrar el proceso. Se compone de una zona estática y otra dinámica.

entrada en la tabla de procesos: define el estado de un proceso. Esta información está siempre disponible para el sistema operativo.

área U (usuario): información de control que se necesita solo en el contexto del proceso

tabla de regiones de proceso: contiene traducción de direcciones virtuales a físicas. También contien un campo con los permisos del proceso.

pila del kernel

## Entrada en la tabla de procesos

estado del proceso.

punteros: a zona U y zona de memoria del proceso (código, datos, pila).

tamaño del proceso: útil al kernel, para saber cuanto espacio asignarle.

identificadores de usuario: ID real e ID efectivo.

identificadores de proceso: PID y PPID.

descriptor de suceso: válido cuando el proceso está dormido.

prioridad: usada en la planificación.

señal: enumera señales no procesadas enviadas a un proceso.

temporizadores: tiempo de ejecución del proceso.

enlace-P: puntero al siguiente enlace enla cola ready.

estado en la memoria: indica sila imagen del proceso está en memoria o en disco.

**Area U**

Puntero a la tabla de procesos.

Ids de usuario real y efectivo.

temporizadores: registro del tiempo que el proceso y sus hijos dedicaron en modo usuario y modo kernel.

vector de señales: indica la acción a tomar ante cada señal definida en el sistema.

terminal de control: tty

campo de error: errores producidos durante una llamada al sistema.

valor de retorno: resultado de las llamadas al sistema.

parámetros de I/O: cantidad de datos a transferir, dirección de origen y destino.

parámetros de archivos: directorio actual y directorio raíz.

tabla de descriptores de archivo: registro de los archivos que el proceso tiene abiertos.

campos de límites: restringe tamaño del proceso y tamaño de archivo que puede crear.

campos de modo de protección: máscara con los modos de protección de los archivos creados por el proceso.

## Control de procesos

La creación de procesos se realiza pormedio de la llamada al sistema fork(), en su invocación ocurre lo siguiente:

1.- Asigna una entrada en la tabla de proceso al nuevo proceso.

2.- Asigna PID al nuevo proceso.

3.- Hace una copia de la imagen del proceso padre, a excepción de la memoria compartida.

4.- Inrementa los contadres de los archivos que son propiedad del padre.

5.- Pone al hijo en el estado ready.

6.- Retorna al padre el PID del hijo y retorna un 0 al hijo.

Durante estas acciones el proceso padre está en modo núcleo.