caracteres y pasar el corrector ortográfico dentro del mismo proceso. Muchos sistemas operativos modernos han ampliado el concepto de proceso para permitir que un proceso tenga múltiples hebras de ejecución y, por tanto, pueda llevar a cabo más de una tarea al mismo tiempo. El Capítulo 4 se ocupa en detalle del análisis de los procesos multihebra.

3.2 Planificación de procesos

El objetivo de la multiprogramación es tener en ejecución varios procesos al mismo tiempo con el fin de maximizar la utilización de la CPU. El objetivo de los sistemas de tiempo compartido es conmutar la CPU entre los distintos procesos con tanta frecuencia que los usuarios puedan interactuar con cada programa mientras éste se ejecuta. Para conseguir estos objetivos, el **planificador de pro-**

REPRESENTACIÓN DE UN PROCESO EN LINUX

El bloque de control de un proceso en el sistema operativo Linux se representa mediante la estructura C `task.struct`. Esta estructura contiene toda la información necesaria para representar un proceso, incluyendo el estado del proceso, la información de planificación y de gestión de memoria, la lista de archivos abiertos y los punteros al padre del proceso y a cualquiera de sus hijos.

```
pid_t pid; /* identificador del proceso */
long state; /* estado del proceso */
unsigned int time_slice /* información de planificación */
struct task_struct *parent; /* padre del proceso */
struct list_head children; /* hijos del proceso */
struct files_struct *files; /* lista de archivos abiertos */
struct mm_struct *mm; /* espacio de direcciones de este proceso */
```

En esta estructura, por ejemplo, el estado de un proceso se representa mediante el campo `long state`. Dentro del *kernel* de Linux, todos los procesos activos se representan mediante una lista doblemente enlazada de objetos `task_struct`, y el *kernel* mantiene un puntero al proceso que se esté ejecutando en el sistema en ese momento. Esto se muestra en la Figura 3.5.

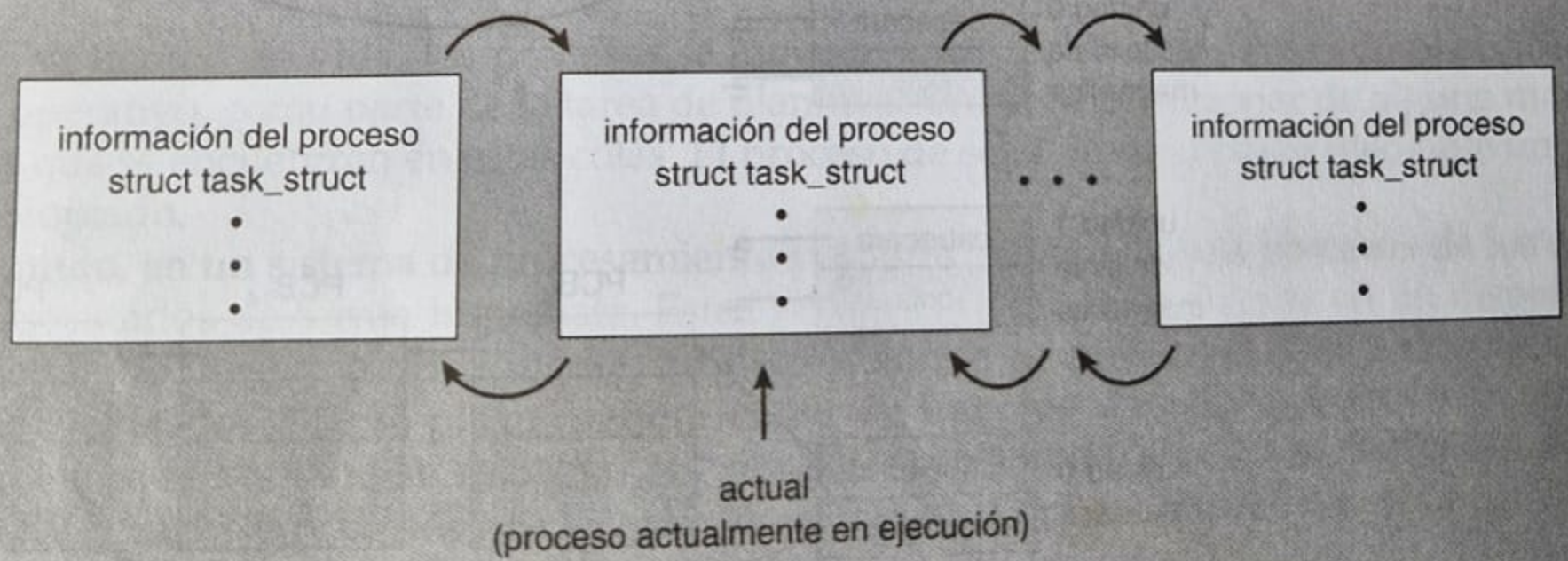

Figura 3.5 Procesos activos en Linux.

Como ilustración del modo en que el *kernel* manipula uno de los campos de `task_struct` para un proceso determinado, vamos a suponer que el sistema desea cambiar el estado del proceso actualmente en ejecución al valor `new_state`. Si `current` es un puntero al proceso que se está ejecutando actualmente, su estado se cambia del siguiente modo:

```
current->state = new_state;
```