# Cosas de ZeOS - Parte 2

## Estructuras de datos

PCB (task\_struct) se implementa en sched.h

Se utiliza un vector de task\_struct para guardar la información de varios procesos.

La pila de sistema se integra dentro de cada task\_struct.

La free\_queue se usa para linkar todas las tasks que hay disponibles para nuevos procesos.

La ready\_queue se usa para linkar todas las tasks candidatas para usar la CPU.

Se puede usar el tipo de datos **union** en C para declarar el task\_union. (pag 37) --> Es la página de 4K

Guardaremos las tasks\_union en un vector de tamaño fijo, llamado tasks. La entrada 0 tendrá el proceso idle.

La free\_queue se tiene que inicializar durante la inicialización del sistema. Se implementa con un struct list\_head (en list.h) (pag 38)

La ready\_queue también es de tipo struct list\_head, y se llamará readyqueue.

Para obtener la dirección de inicio del task\_struct, tenemos que hacer una mascara de esp con los últimos X bits a 0, siendo X el numero de bits usados por el tamaño de la pila. Podemos hacer una función current() que nos devuelva el tastk\_struct actual.

## Gestión de memoria

Todos los procesos de ZeOS acceden a las mismas páginas lógicas.

**Direcciones lógicas:** unsigned int de 32 bits. Se traducen por la MMU.

**Direcciones físicas:** unsigned int de 32 bits. Se usan para acceder a lugares de memoria del chip en sí.

**Mecanismo de paginación:** La memoria se divide en frames del mismo tamaño. La TP traduce de lógica a física. Hay que **activar el bit PG del registro cr0!!!** Si PG = 0 significa que lógica y físicas son iguales.

**Directorio y Tabla de Páginas:** El directorio contiene la dirección base de las Tablas de páginas definidas para un proceso. Las TP contienen la traducción de direcciones lógicas a físicas. El registro **cr3** contiene la dirección fisica base para el código del kernel.

Los bits 31-22 definen la entrada al **directorio**.

Los bits 21-12 definen la **página lógica** e indexa la TP.

Los bits 11-0 definen el **offset** de la página (hasta 4K).

Por lo tanto, page\_number = (address >> 12)

Todo esto de los niveles sirve para encontrar espacio para las TP según sea necesario. En ZeOS solo necesitamos una TP por proceso. Por lo tanto, el directorio solo tiene una entrada válida (la entrada 0).

Es el SO el responsable de cargar la dirección base del directorio del proceso running en el **cr3**. Por eso, ZeOS guarda la dirección vase del directorio en su PCB (campo dir\_pages\_baseAddr del task\_struct).

El vector dir\_pages guarda el directorio, y el vector pagusr\_table la TP. (En mm.c). Estos vectores tienen NR\_TASKS entradas, y cada entrada está linkada a un task\_struct del vector tasks, a través del campo dir\_pages\_baseAddr.

init\_dir\_pages del archivo mm.c contiene la implementación de estas estructuras.

Una entrada de la tabla de páginas es como la de la **página 42 del PDF**.