tss->culo = abierto

1) Crear una nueva syscall
   En primer lugar creamos una entrada en la IDT en el offset del número de interrupción que elegimos para la syscall. El selector tendrá DPL nivel 3 para que las tareas puedan accederlo. Este selector apunta a una entrada de la GDT que tiene DPL 0 de modo que solo el kernel al atender la interrupción pueda usarla, y apunta a la RAI.

   Asumimos que ya todas las tareas tienen definidas las entradas correspondientes en la GDT para los segmentos de código, datos y su TSS.

2) Crear una nueva tarea
   Llamar a **create_task()**. Recibe un tipo de tarea (0 o 1). Ahí adentro se busca una entrada disponible en la gdt, se le crea una tss (donde se inicia el esquema de paginación, y se inician las direcciones virtuales de código y stack). Devuelve el task_id.

3) Pasar información de una tarea a otra
   Asumiendo que cada tarea ya fue interrumpida una vez por el clock.

1) Por medio de registros. Tarea A quiere pasarle un número a tarea B. Lo guarda en registro de prop general. Al momento de ser interrumpida el clock, mientras no se haga ningun call y el registro no sea modificado, en la rutina hay que:
   a) Buscar la tss de la tarea B
   b) Buscar el esp de la tarea B
   c) Usando offset, modificar el valor del registro deseado, haciendo mov [esp + offset], registro. El offset si solo se hizo pushad está definido en el archivo del taller, si antes se pusheo otra cosa hay que tenerlo en cuenta. Cada registro ocupa 4 bytes.
   d) Al entrar la tarea B y hacer el popad, se guarda en el registro de la tarea B

2) Si no se utilizan registros de prop general, se puede utilizar la memoria compartida. Para esto podriamos modificar el struct de environment, y luego las tareas pueden acceder haciendo environment->campo_nuevo. La shared es readonly para las tareas, solo la modifica el kernel.

4) Habilitar/Deshabilitar tarea
   El scheduler tiene la función sched_disable_task, que recibe un task_id y la deshabilita je.

5) OPcode invalido = interrupción numero 6, no pushea error code, el eip apunta a la instrucción que generó la excepcion.

6) General protection = interrupción 13. La levanta por ejemplo HALT cuando la llama una tarea de nivel 3. Pushea error code, el eip apunta a la instrucción que generó la excepción

7) Llegar desde selector a la tss= (referencia, parcial de ***DIOS***)
    - Shifteamos el selector 3 bits a la derecha(limpiar atributos).
    - Indexamos en la gdt -> gdt[index shifteado]
    - Reconstruir direccion:
    -

```
get_tss_direction(gdt[task_sel >> 3]); // Llamada a funcion

tss* get_tss_direction(gdt_entry_t selector){ // es el selector ya shifteado
    uint32_t base_addr = selector.base_31_24 << 24 |
                         selector.base_23_16 << 16 |
                         selector.base_16_0;

    return (tss*) base_addr;
}
```

COSAS QUE YA TENEMOS EN EL TALLER:
4 descriptores en la GDT: (código y datos) x (nivel 0 y nivel 3).
1 descriptor en la GDT de segmento de video
Interrupciones 32 y 33 (reloj y teclado) (si redefinimos una del 1 a 14, borramos la línea de arriba).

PADRE NUESTRO QUE ESTÁS EN EL CIELO SANTIFICADO SEA TU NOMBRE VENGA A NOSOTROS TU REINO, HAGASE SU VOLUNTAD EN LA TIERRA COMO EN EL CIELO, DANOS HOY TU PAN DE CADA DIA, PERDONA NUESTRAS OFENSAS COMO NOSOTROS PERDONAMOS A LOS QUE NOS OFENDEN , NO NOS DEJES CAER EN LA TENTACION, MAS LIBRANNOS DEL MAL, AMEN
HACEME APROBAR ORGA2

Tenemos tss_tasks -> array de tss, indexa por task_id