Task State Segment

From OSDev Wiki

The Task State Segment (TSS) is a special data structure for x86 processors which holds information about a task. The TSS is primarily suited for hardware multitasking, where each individual process has its own TSS. In Software multitasking, one or two TSS's are also generally used, as they allow for entering ring0 code after an interrupt.

Contents

Structure

offset	31-16	15-0
0x00	reserved	LINK
0x04	ESP0	
0x08	reserved	SS0
0x0C	ESP1	
0x10	reserved	SS1
0x14	ESP2	
0x18	reserved	SS2
0x1C	CR3	
0x20	EIP	
0x24	EFLAGS	
0x28	EAX	
0x2C	ECX	
0x30	EDX	
0x34	EBX	
0x38	ESP	
0x3C	EBP	
0x40	ESI	
0x44	EDI	

0x48	reserved	ES
0x4C	reserved	CS
0x50	reserved	SS
0x54	reserved	DS
0x58	reserved	FS
0x5C	reserved	GS
0x60	reserved	LDTR
0x64	IOPB offset	reserved

TSS in software multitasking

For each CPU which executes processes possibly wanting to do system calls via interrupts, one TSS is required. The only interesting fields are SS0 and ESP0. Whenever a system call occurs, the CPU gets the SS0 and ESP0-value in its TSS and assigns the stack-pointer to it. So one or more kernel-stacks need to be set up for processes doing system calls. Be aware that a thread's/process' time-slice may end during a system call, passing control to another thread/process which may as well perform a system call, ending up in the same stack. Solutions are to create a private kernel-stack for each thread/process and re-assign esp0 at any task-switch or to disable scheduling during a system-call.

Setting up a TSS is straight-forward. An entry in the Global Descriptor Table is needed (see also the GDT Tutorial), specifying the TSS' address as "base", TSS' size as "limit", 0x89 (Present|Executable|Accessed) as "access byte" and 0x40 (Size-bit) as "flags". In the TSS itself, the members "SS0", "ESP0" and "IOPB offset" are to be set:

- SS0 gets the kernel datasegment descriptor (e.g. 0x10 if the third entry in your GDT describes your kernel's data)
- ESP0 gets the value the stack-pointer shall get at a system call
- IOPB may get the value sizeof(TSS) (which is 104) if you don't plan to use this io-bitmap further (according to mystran in http://forum.osdev.org/viewtopic.php?t=13678)

The actual loading of the TSS must take place in protected mode and after the GDT has been loaded. The loading is simple as:

```
mov ax, 0x??  ;The descriptor of the TSS in the GDT (e.g. 0x28 if the si
ltr ax        ;The actual load
```

See Also

- GDT Tutorial
- System Calls
- Getting to Ring 3

Threads

- Do I need a TSS?

External Links

- Task State Segment on Wikipedia

Retrieved from "http://wiki.osdev.org/index.php?title=Task_State_Segment&oldid=11145"
Category: X86 CPU

- This page was last modified on 5 February 2011, at 06:21.
- This page has been accessed 35,910 times.