

Process

- 시스템 메모리에서 실행 중인 프로그램
- 스케줄링 대상인 태스크

❖ 프로세스는 어떤 구조체로 식별되는가?

- 프로세스 관리 → Task Descriptor 자료구조
- 최상단에 thread Info 구조체 존재

```
struct task_struct {  
    struct thread_info   
        //프로세스 ID 등  
        //필요한 정보들..  
}
```

Pointer
Process State
Process Number (PID)
Program Counter
Registers
Memory Limits
Open File Lists
...

❖ Task 란?

- Process 라고 봐도 무방

Process

Process의 자료구조

```
(struct task_struct *) (struct task_struct*) 0xf1618000 = 0xF1618000
  · (long int) state = 1 ≈ 0x1 ≈ 'NNN',  

  ⊕ (void *) stack = 0xF1604000 ≈ __key+0x2F721D2C → ,  

  ⊕ (atomic_t) usage = ((int) counter = 2 ≈ 0x2 ≈ 'UUU'),  

  · (unsigned int) flags = 1077936384 ≈ 0x40400100 ≈ '@@SN',  

  · (unsigned int) ptrace = 0 ≈ 0x0 ≈ 'NNNN',  

  ⊕ (struct llist_node) wake_entry = ((struct llist_node *) next = 0x0),  

  · (int) on_cpu = 0 ≈ 0x0 ≈ 'UUU',  

  ⊕ (struct task_struct *) last_wakee = 0xC2577400 ≈ __key+0x69512C →
```

Process의 stack 상단 주소
→ stack pointer

Process의 Thread Info 구조체 :: stack을 Thread info로 casting

```
⊕ (struct thread_info *) (struct thread_info*) 0xF1604000 = 0xF1604000 ≈ __key+0x2F721D2C → (
  · (long unsigned int) flags = 0 ≈ 0x0 ≈ 'NNNN',  

  · (int) preempt_count = 2 ≈ 0x2 ≈ 'UUU',  

  · (mm_segment_t) addr_limit - 3201448256 ≈ 0xBFFFFFFF ≈ 'BNNN',  

  ⊕ (struct task_struct *) task = 0xF1618000 ≈ __key+0x2F735D2C → ((long int) state = 1 ≈ 0x1 ≈ 'NN',  

  ⊕ (struct exec_domain *) exec_domain = 0xC1A1AFDC ≈ default_exec_domain → ((char *) name = 0xC145
  · (u32) cpu = 2 ≈ 0x2 ≈ 'NNNS',  

  · (u32) cpu_domain = 0 ≈ 0x0 ≈ 'NNNN',  

  ⊕ (struct cpu_context_save) cpu_context = ((u32) r4 = 4056721280 ≈ 0xF1CCA780 ≈ 'FCA8', (u32)
  · (u32) syscall = 0 ≈ 0x0 ≈ 'UUUU',  

  ⊕ (u8 [16]) used_cp = "",  

  ⊕ (long unsigned int [2]) tp_value = ([0] = 1160496 ≈ 0x0011B530 ≈ 'U150', [1] = 0 ≈ 0x0 ≈ 'NNNN')
  ⊕ (union fp_state) fpstate = ((struct fp_hard_struct) hard = ((unsigned int [35]) save = ([0] = 0
  ⊕ (union vfp_state) vfpstate = ((struct vfp_hard_struct) hard = ((u64 [32]) fpregs = ([0] = 2911
```

- Thread info → Process 참조 :: 따라서, Thread info가 자신이 포함된 Process 확인 가능

Thread Info 자료구조

```
(struct thread_info *) (struct thread_info*)0xF1604000 = 0xF1604000 ≈ __key+0
  • (long unsigned int) flags = 0 ≈ 0x0 ≈ 'NNNN' _UUUU',
  • (int) preempt_count = 2 ≈ 0x2 ≈ 'NNNS' _UUUX',
  • (mm_segment_t) addr_limit = 3204448256 ≈ 0xBF000000 ≈ 'BNNN' _FUUU',
  • (struct task_struct *) task = 0xF1618000 ≈ __key+0x2F735D2C → ((long int))
  • (struct exec_domain *) exec_domain = 0xC1A1AFDC ≈ default_exec_domain → ((0
  • (__u32) cpu = 2 ≈ 0x2 ≈ 'NNNS' _UUUX',
  • (__u32) cpu_domain = 0 ≈ 0x0 ≈ 'NNNN' _UUUU',
  • (struct cpu_context_save) cpu_context = (
    • (__u32) r4 = 4056721280 ≈ 0xF1CCA780 ≈ 'FCAB' _1C70',
    • (__u32) r5 = 4049698816 ≈ 0xF1618000 ≈ 'F8H' _1ad0U',
    • (__u32) r6 = 3842940160 ≈ 0xE50E9D00 ≈ 'E9N' _5ODU',
    • (__u32) r7 = 3932291520 ≈ 0xEA6201C0 ≈ 'EbSC' _AbH0',
    • (__u32) r8 = 3248547128 ≈ 0xC1A0E538 ≈ 'CAE' _1038',
    • (__u32) r9 = 3880364992 ≈ 0xE749ABC0 ≈ 'EIA' _7IB0',
    • (__u32) s1 = 4049699952 ≈ 0xF1618470 ≈ 'Fa' _1a+p',
    • (__u32) fp = 4049624708 ≈ 0xF1605E84 ≈ 'F' _1A4',
    • (__u32) sp = 4049624640 ≈ 0xF1605E40 ≈ 'F' _1A@',
    • (__u32) pc = 3237955160 ≈ 0xC0FF4658 ≈ 'CF' _0FFX'
  • (__u32 [2]) extra = ([0] = 0 ≈ 0x0 ≈ 'UUUU', [1] = 0 ≈ 0x0 ≈ 'NNNN' _UUUU')),
```

- Thread Info의 CPU Context field에 Register 값들을 저장해 놓았다.
- 즉, CPU Context란 Register의 값들이다.