

# Linux Scheduler Architecture

# Hierarchical scheduling class in Linux

`pick_next_task()` in `kernel/sched/core.c` essentially does this:

```
for (class = sched_class_highest;
     class != NULL;
     class = class->next;)
{
    p = class->pick_next_task(rq);
    if (p)
        return p;
}

// The idle class should always have a runnable task
BUG();
```

## struct sched\_class (up to kernel 5.8)

```
const struct sched_class rt_sched_class = {  
    .next                = &fair_sched_class,  
    .enqueue_task        = enqueue_task_rt,  
    .dequeue_task        = dequeue_task_rt,  
    .yield_task          = yield_task_rt,  
    .check_preempt_curr  = check_preempt_curr_rt,  
    .pick_next_task      = pick_next_task_rt,  
    .put_prev_task       = put_prev_task_rt,  
    .set_next_task       = set_next_task_rt,  
};
```

## struct sched\_class (since kernel 5.9)

```
const struct sched_class rt_sched_class
    __section("__rt_sched_class") = {
    .enqueue_task          = enqueue_task_rt,
    .dequeue_task         = dequeue_task_rt,
    .yield_task           = yield_task_rt,
    .check_preempt_curr   = check_preempt_curr_rt,
    .pick_next_task       = pick_next_task_rt,
    .put_prev_task        = put_prev_task_rt,
    .set_next_task        = set_next_task_rt,
```

# Array of sched\_class

- Sched classes are now arranged in an array by linker scripts

- include/asm-generic/vmlinux.lds.h:

```
#define SCHED_DATA \
    STRUCT_ALIGN(); \
    __begin_sched_classes = .; \
    *(__idle_sched_class) \
    *(__fair_sched_class) \
    *(__rt_sched_class) \
    *(__dl_sched_class) \
    *(__stop_sched_class) \
    __end_sched_classes = .;
```

- for\_class\_range() macros in 5.8:

```
#define for_class_range(class, _from, _to) \
    for (class = (_from); class != (_to); class = class->next)
```

- for\_class\_range() macros in 5.10:

```
#define for_class_range(class, _from, _to) \
    for (class = (_from); class != (_to); class--)
```

# Runqueue data structures

- **struct rq** (kernel/sched/sched.h)
  - Main per-CPU runqueue data structure
  - Contains per-sched\_class runqueues: `cfs_rq`, `rt_rq`, etc.
- **struct sched\_entity** (include/linux/sched.h)
  - `sched_<class>_entity` for each sched\_class (except that `sched_entity` is for cfs)
  - Member of `task_struct`, one per each sched\_class
- **task\_struct.sched\_class**
  - Pointer to the current sched\_class for the task
  - `sched_setscheduler()` syscall changes process's sched\_class