STOS - Scheduling

Gabriel Laskar <gabriel@lse.epita.fr>

Assignment

- On the website (ksto-4)
- Code submitted in branch “k4”
- Still accepting patches
- Due to February 6

Module: scheduler.ko

```
MODINFO {
    module_name("scheduler"),
    module_init(init),
    module_type(M_SCHED),
    module_deps(M_TASK | M_TIMER)
};
EXPORT_SYMBOL(start_scheduling);
EXPORT_SYMBOL(schedule);
EXPORT_SYMBOL(enqueue_task);
EXPORT_SYMBOL(__sleep_on);
EXPORT_SYMBOL(wake_up);
EXPORT_SYMBOL(wake_up_task);
```

First Part: Scheduling

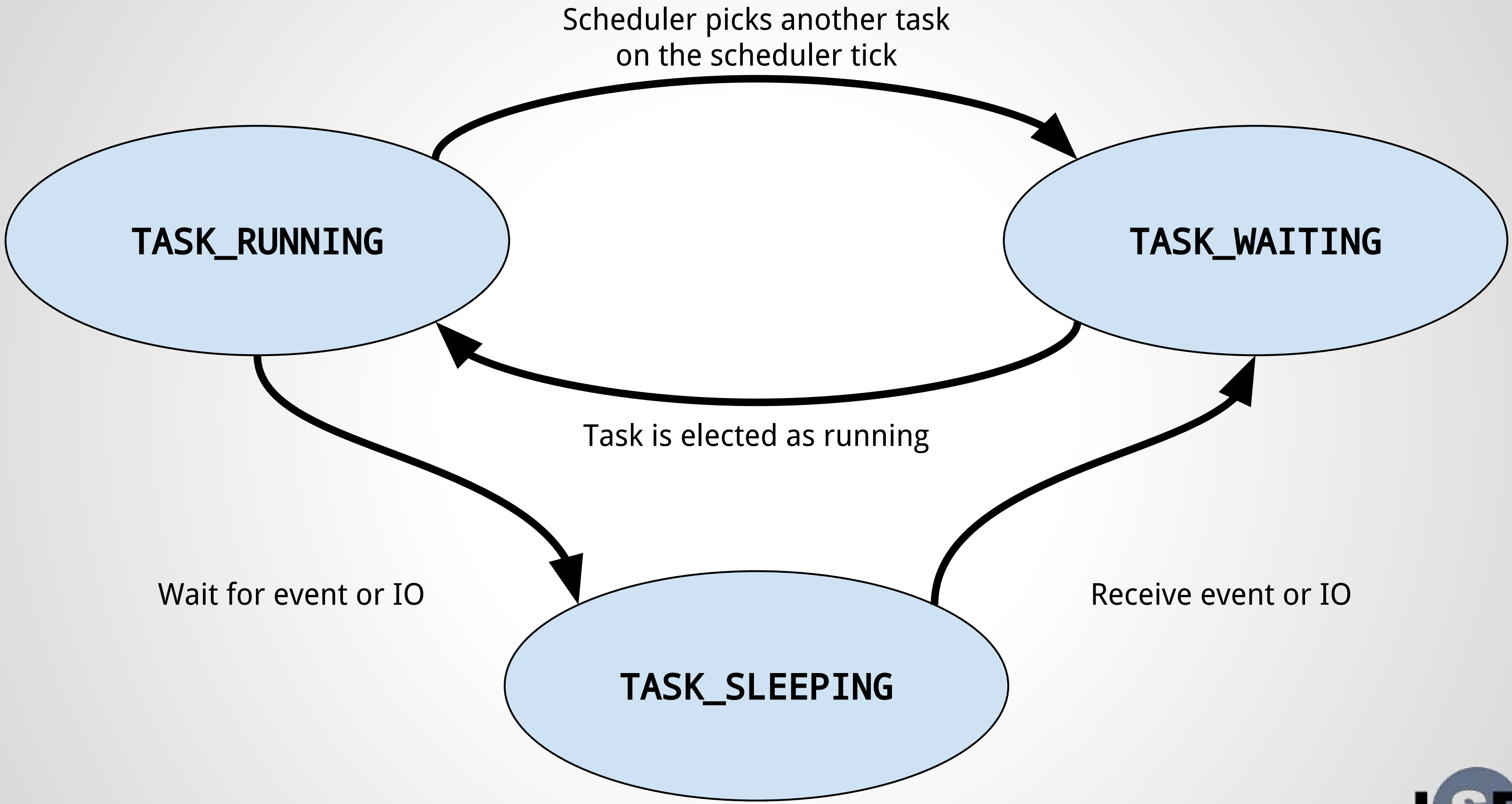
```c
/* Scheduler entry point */
void schedule(void);

/* Set PIT IRQ handler to call schedule() function */
void start_scheduling(void);

void enqueue_task(struct task* t);

void wake_up_task(struct task* t);
```

What’s in a task?

```
struct task {
	volatile enum task_state state;
	pid_t pid;

	uid_t uid;
	gid_t gid;

	/* Double chained list of tasks */
	struct list_node tasks;

	/* Tree of tasks */
	struct task* father;
	struct list_node brothers;
	struct task* oldest_son;

	/* Zombies tasks */
	struct list_node zombies;
	spinlock_t zombies_lock;
	uword kernel_sp;
	uword kernel_ip;

	struct regs* regs;

	struct fs* fs;
	struct filedesc* fds;

	struct mem* mem;

	struct sched_attr* sched_attr;
};
```

Task states

```
enum task_state {
    TASK_RUNNING, /* Running on a CPU */
    TASK_WAITING, /* Waiting for CPU time. */
    TASK_SLEEPING, /* Waiting for a data to be ready
*/
    TASK_IDLING, /* Inside the idle loop. */
    TASK_ZOMBIE
};
```


Scheduler picks another task on the scheduler tick
TASK_RUNNING
TASK_WAITING
Task is elected as running
Wait for event or IO
Receive event or IO
TASK_SLEEPING

Initialize the module

- initialize the sched_attr for the init task and create an idle task

```
struct sched_attr sched_attr = {
     .prio = IDLE_PRIO,
     .running_time = 0,
};

/* Initialize the idle task */
idle_task = clone_task(get_current(), CLONE_FORK);
prepare_new_task(idle_task, idle, NULL, KERNEL_TASK);
```

Second Part: Sleep Queues

```
struct sleep_queue {
	struct list_node task_list;
	spinlock_t queue_lock; /* Protect the linked list */
};

static inline void init_sleep_queue(struct sleep_queue* q) { … }

/* Sleep until the condition becomes true. */
#define sleep_on(q, condition)     \
	do {                           \
		while (!(condition))       \
			__sleep_on(q);         \
	} while (0)

/* Wake up all the process that sleeps on this sleep queue. */
void wake_up(struct sleep_queue* q);

/* Sleep until a wake_up. This is not expected to be used as such. */
void __sleep_on(struct sleep_queue* q);
```

wake_up()

```
void wake_up(struct sleep_queue* q)
{
    struct task* t;
    lfor_each_entry_safe(&q->task_list, t,
tasks)
        wake_up_task(t);
}
EXPORT_SYMBOL(wake_up);
```

__sleep_on()

- set task to state `TASK_SLEEPING`
- remove the task from the active ones and put it in the sleep queue.
- call `schedule()` in order to schedule another task