RTX API

release_processor

```
int release_processor();
```

- **returns**: `RTX_OK` if successful, otherwise `RTX_ERR`

Releases the current process from the processor. The next queued process is then returned from the scheduler and switched in as the current process.

set_process_priority

```
int set_process_priority(int process_id, int priority);
```

- **process_id**: ID of the process to set
- **priority**: priority in [0,3] to set
- **returns**: `RTX_ERR` if priority or ID are invalid, otherwise `RTX_OK`

Sets the priority of the given process. If the new priority is greater than the priority of the currently running process, the current process will be preempted. The null process and interrupt processes cannot be set.

get_process_priority

```
int get_process_priority(int process_id);
```

- **process_id**: ID of the process to get
- **returns**: the priority of the given process

Gets the priority of the given process.

request_memory_block

```
void* request_memory_block();
```

- **returns**: a pointer to the next memory block from the heap, or `NULL` if no memory blocks are available

Retrieves a memory block. If there are no memory blocks remaining, the current process state will be switched to `BLOCKED` and released from the processor.

release_memory_block

```
int release_memory_block(void* memory_block);
```

- **memory_block**: a pointer to the memory block to release
- **returns**: `RTX_OK` if successful, otherwise `RTX_ERR`

Restores a memory block to the heap. The memory block becomes available for use if requested. If a process with a higher priority than the current process is blocked, that process will preempt the current process and will be given the released memory block.