RTX API

release_processor

```
int release_processor();
```

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

The function will return `0` on a successful operation and `1` otherwise.

set_process_priority

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

Sets the priority of the given process. The currently running process will be taken out of the processor and replaced with the given process if the given process has a higher priority. The null process cannot be set.

The function will return `0` on a successful operation and `1` otherwise (invalid priority).

get_process_priority

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

Gets the priority of the given process.

The function will return `0` on a successful operation and `1` otherwise (invalid priority).

request_memory_block

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

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

The function will return the memory block pointer, which will also be `NULL` if there are no free memory blocks.

release_memory_block

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

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.

The function will return `0` on a successful operation and `1` otherwise.