Схема работы скедулера с кражей задач (Work stealing scheduler)

Идея: если в нашем пуле закончились задачи, то будем искать их в других пулах

1 пул (ABT_pool) ассоциируется с одним планировщиком (ABT_sched).
Пусть каждый планировщик знает про все пулы (вообще все). Постараемся их занумеровать. Для этого будем использовать обход по часовой стрелке: текущий пул имеет номер 0; следующий за ним - номер 1 и т.д. (подробнее на картинках). Иными словами: расстояние от заданного пула при обходе по часовой стрелке. Будем называть это относительной нумерацией. Абсолютная нумерация - номер пула в относительной нумерации пула номер 0.

Рисунок 1. Мир глазами планировщика пула номер 0. Демонстрирует абсолютный и относительный порядок (которые в данном случае совпадают) и направление обхода

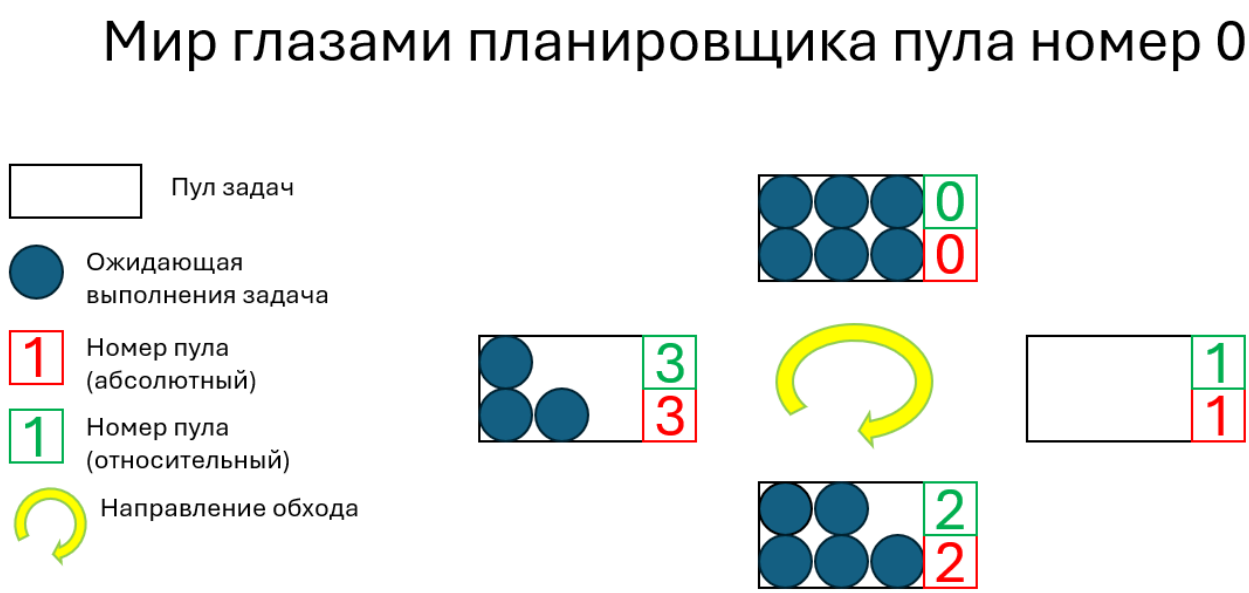
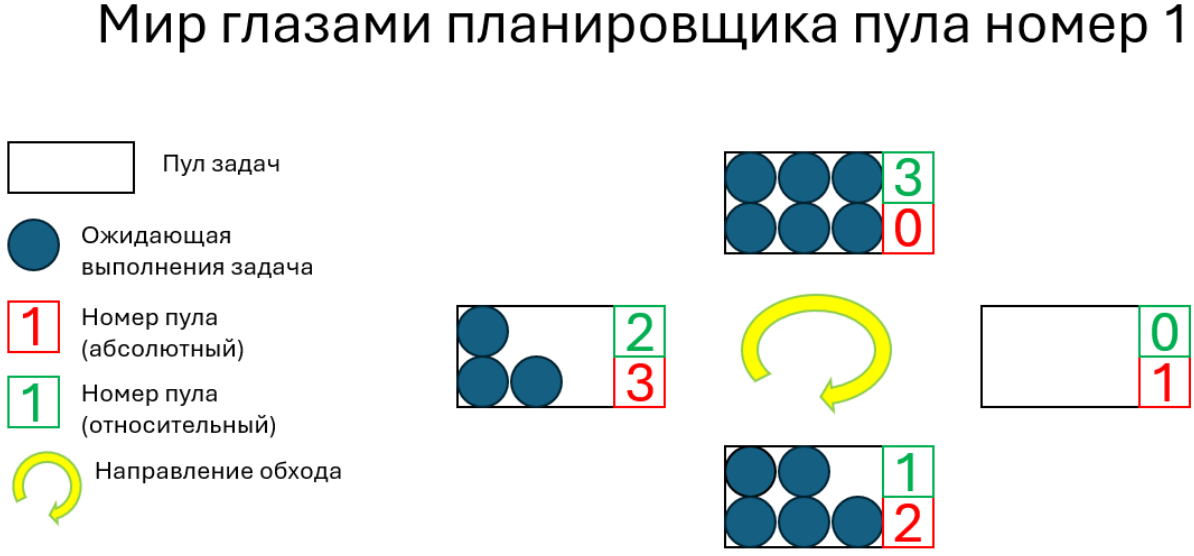


Рисунок 2. Мир глазами планировщика пула номер 1. Демонстрирует абсолютный и относительный порядок и направление обхода в случае планировщика номер 1



Алгоритм работы планировщика

1. Пытаемся взять в работу задачу из своего пула задач. Если задача есть, то выполняем её;
2. Если задачи нет, то идём по часовой стрелке пока не найдём свободную задачу из другого пула. Если нашли задачу –
3. Если в результате прошли целый круг и не нашли задачу, то ничего не делаем.

Описание работы планировщика в картинках

Рисунок 3. Начальное состояние. Планировщик номер 1 выполнил все свои задачи

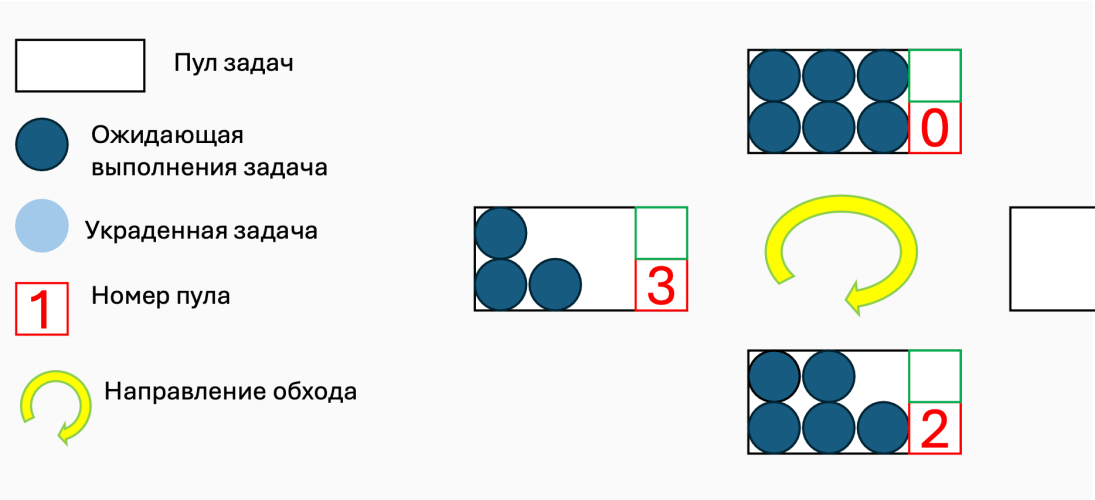


Рисунок 4. Планировщик номер 1 выполняет обход по часовой стрелке и находит задачу в соседнем пуле номер 2

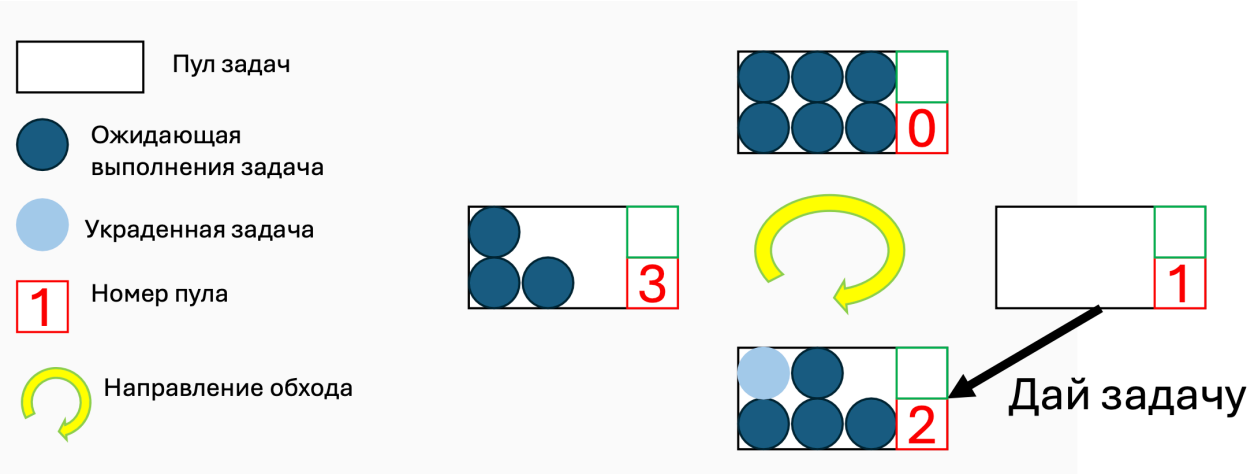
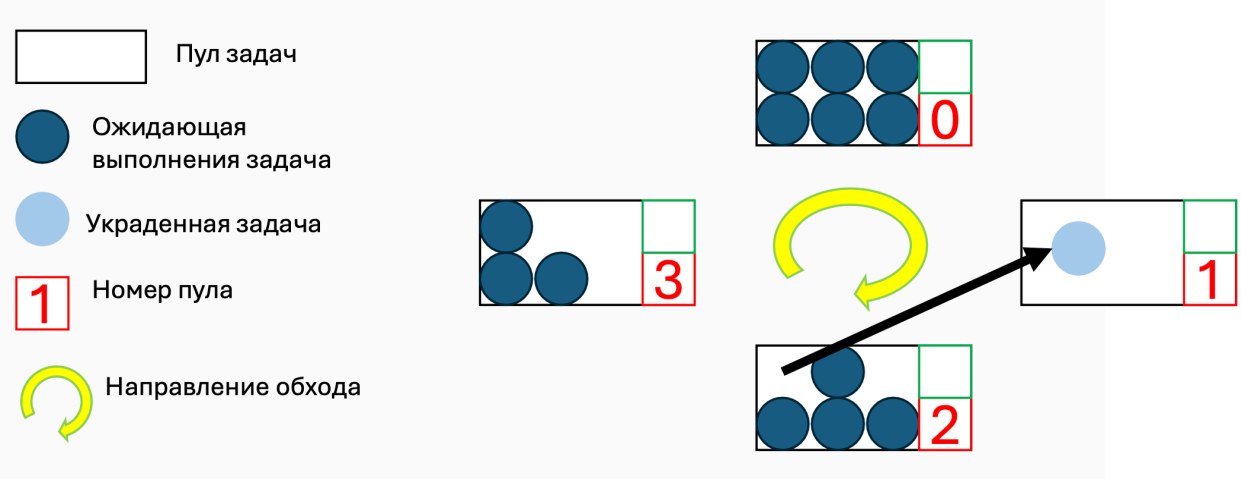


Рисунок 5. Планировщик номер 1 крадёт задачу из пула номер 2



Технические детали реализации

```
/// Интерфейс для пользователя (.h файл)
Внутри себя задаёт конфигурацию для скедулеров, а также распределяет их по кругу (тем самым определяя относительный порядок)
void ABT_create_ws_scheds(int num, ABT_pool *pools, ABT_sched *scheds);
/// Детали реализации, скрыты от пользователя (.c файл)
Структура для задания информации (data) внутри скедулера
typedef struct { … } ws_sched_data_t;
Устанавливает data (ABT_sched_set_data) для скедулера
static int sched_init(ABT_sched sched, ABT_sched_config config)
Обратное действие к sched_init – затирает data (освобождает память) для скедулера
static int sched_free(ABT_sched sched)
Функция, в которой и будет находиться “сердце” скедулера (алгоритм описанный выше)
static void sched_run(ABT_sched sched)
```