Assignment 2
Operating System

Anikait Sahota
2018016

---

* The code corresponding to push_back :

```
static void push_back(struct thread *t)
{
	if(t == NULL)    return ;
	if(ready_list == NULL) {
		ready_list = t ;//  t->next = NULL ;
		return ;
	}
	struct thread* thrd = ready_list ;
	while (thrd->next != NULL)
		thrd = thrd->next ;
	thrd->next = t ;
	t->prev = thrd ;//t->next = NULL ;
}
```

---

* The code corresponding to pop_front :

```
static struct thread *pop_front()
{
	if(ready_list == NULL)   return NULL ;
	struct thread* thrd = ready_list ;
	ready_list = ready_list->next ;
	if(ready_list != NULL)
		ready_list->prev = NULL ;
	if(thrd->next == NULL) ready_list = NULL ;
	thrd->next = NULL ;
	return thrd ;
}
```

---

* code corresponding to create_thread :

```
void create_thread(func_t func, void *param)
{
	struct thread* new_thrd = malloc(sizeof(struct thread)) ;
	unsigned* stack = (malloc(PAGE_SIZE) + PAGE_SIZE);

	stack-- ; *(func_t*)stack = param ;
	stack-- ; *stack = 0 ;
	stack-- ; *(func_t*)stack = func ;
	stack-- ; *stack = 0 ;
	stack-- ; *stack = 0 ;
	stack-- ; *stack = 0 ;
	stack-- ; *stack = 0 ;

	new_thrd->esp = stack ;
	new_thrd->next = NULL ;
	new_thrd->prev = NULL ;
	push_back(new_thrd) ;    // pushing the new_thrd in ready_list
}
```

---

* This is the make test output :

```
starting main thread: num_threads: 1024
main thread exiting: counter:30768239300147200
./app 1024 1
starting main thread: num_threads: 1024
bar1: (nil)
bar2: (nil)
bar1: 0x1
main thread exiting: counter:0
```

---

* We can implement the thread_exit in such way, that when thread exits, it frees the structure also

```
void thread_exit()
{
	if(cur_thread != NULL)
	{
		free(cur_thread->esp) ;
		free(cur_thread) ;
		cur_thread = NULL ;
	}
	schedule() ;
}
```