Akshala Bhatnagar
2018012

OS Assignment-2

1. Paste your code corresponding to push_back

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

2. Paste your code corresponding to pop_front

```
static struct thread *pop_front()
{
    if(ready_list == NULL){
        return NULL;
    }
    else if(ready_list->next == ready_list){
        struct thread * returned_thread = ready_list;
        ready_list = NULL;
        return returned_thread;
    }
```

Akshala Bhatnagar
2018012

```
        else{
            struct thread *returned_thread = ready_list;
            struct thread *ready_list_prev = ready_list->prev;
            struct thread* ready_list_next = ready_list->next;
            ready_list = ready_list_next;
            ready_list->prev = ready_list_prev;
            ready_list_prev->next = ready_list;
            return returned_thread;
        }
    }
```

3. Paste your code corresponding to create_thread. If you are calling functions that are defined by you in create_thread, paste the code of them as well.

```
void create_thread(func_t func, void *param)
{
    unsigned *stack = malloc(4096);
    stack += 1024;
    struct thread* created_thread =  malloc(sizeof(struct thread));
    stack -= 1;

    *(void**)(stack) = param;
    *(stack-1) = 0;
    *(func_t*)(stack-2) = func;
    *(stack-3) = 0;
    *(stack-4) = 0;
    *(stack-5) = 0;
    *(stack-6) = 0;

    created_thread->esp = stack-6;
    push_back(created_thread);
}
```

4. Dump the output of the‘‘make test’’.

```
./app 1024
```

starting main thread: num_threads: 1024
main thread exiting: counter:307682393001472 00
./app 1024 1
starting main thread: num_threads: 1024
bar1: (nil)
bar2: (nil)
bar1: 0x1
main thread exiting: counter:0

5. Suggest a strategy to free struct thread and the stack corresponding to the thread that has exited (using thread_exit API). You don't need to implement this logic.

   We can create a garbage_list that consists of all threads that have exited. In the schedule function, we clear this garbage list before calling context_switch. After clearing the list, the cur_thread is put in the garbage_list and then context_switch is called.