```
#ifndef __MTHREAD_MTHREAD_INTERNAL_H__
#define __MTHREAD_MTHREAD_INTERNAL_H__
#ifdef __cplusplus
extern "C"
{
#endif


#define TWO_LEVEL


#include <stdlib.h>
#include <stdio.h>
#include <ucontext.h>

#ifndef __GNUC__
#define inline
#endif

#include "mthread.h"

typedef struct {
  volatile struct mthread_s* first;
  volatile struct mthread_s* last;
  mthread_tst_t lock;
}mthread_list_t;

typedef struct {
  struct mthread_s* idle;
  volatile struct mthread_s* current;
  mthread_list_t ready_list;
  int rank;
  volatile int state;
  volatile struct mthread_s* resched;
}mthread_virtual_processor_t;

typedef enum{RUNNING,BLOCKED,ZOMBIE} mthread_status_t;

struct mthread_s{
  ucontext_t uc;
  volatile void * res;
  void* arg;
  void *(*__start_routine) (void *);
  volatile struct mthread_s* next;
  volatile mthread_status_t status;
  void* stack;
};

#define MTHREAD_LIST_INIT {NULL,NULL,0}

  extern int mthread_test_and_set(mthread_tst_t *atomic);
  extern void mthread_spinlock_lock(mthread_tst_t *atomic);
  extern void mthread_spinlock_unlock(mthread_tst_t *atomic);
  extern int mthread_get_vp_rank();
```

```
  extern void __not_implemented (const char *func, char *file, int line);
  extern void *safe_malloc(size_t size);
  extern int mthread_log(char* part, const char *format, ...);
  extern int mthread_log_init();

  extern void mthread_insert_first(struct mthread_s* item, mthread_list_t* list);
  extern void mthread_insert_last(struct mthread_s* item, mthread_list_t* list);
  extern struct mthread_s* mthread_remove_first(mthread_list_t* list);

  extern void __mthread_yield(mthread_virtual_processor_t* vp);
  extern mthread_virtual_processor_t* mthread_get_vp();
#define not_implemented() __not_implemented(__FUNCTION__,__FILE__,__LINE__)

#ifdef __cplusplus
}
#endif
#endif
```