

CW 1

Raul Lopes

March 30, 2001

Implement a *CPU scheduler* for either the *Solaris* or the *Linux* OS. Choose one of the following options:

Priority Scheduler: 80/100

Multilevel Scheduler: 90/100, if your scheduler has both *roundrobin* and *SJF*.

Multilevel with feedback: 90/100, if your scheduler has both *roundrobin* and *SJF*.

An extra 10/100 will be given for a package offering lock/unlock primitives, implemented using the Bakery algorithm. Neither signals, nor semaphores should be used in this item.

1 Your package

Your package will be implemented as a module named

mthread.c

with interface defined in

mthread.h

Your package will have at least the following functions:

```
void mthread_init(); // used to initialize the thread system
```

```
void mthread_finalize(); // used to finalize the thread system
```

```

int mthread_create(void *(*f) (void *a), void* v, int p, int *t)
    // used to create a thread, receives
    // f, pointer to function
    // v, pointer to argument
    // p, priority
    // t, pointer to area that will receive id

void mthread_exit(void *) // finishes thread and returns result

```

Additionally, it could implement mutex primitives:

```

int m_lock_init(m_lock_t *); // initializes lock
int m_lock(m_lock_t *); // lock
int m_unlock(m_lock_t *t); // unlock

```

1.1 Restrictions

Your package will permit the creation of up to 128 threads in up to 16 levels of priority.

2 Monitor

Design and implement a framework for monitors. Use the structure implemented in your *mthread* package. This could be worse 11 in 10.

3 The rules of game

Deadline: 2/apr/2001, 18h.

Working in groups: *SOR*.

Plagiarism: *SOR*.

Documentation: *SOR*.