Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

Faculty of Engineering, Mathematics and Science

School of Computer Science & Statistics

Integrated Computer Science **Trinity Term 2017**
Year 2 Annual Examinations

Concurrent Systems and Operating Systems

Thursday 11 May 2017 **RDS** **14:00 – 16:00**

Dr Mike Brady

Instructions to Candidates:

Attempt **two** questions. All questions carry equal marks. Each question is scored out of a total of 20 marks.

You may not start this examination until you are instructed to do so by the Invigilator.

Materials permitted for this examination:

A two-page document, entitled *"Pthread Types and Function Prototypes"* accompanies this examination paper.

Non-programmable calculators are permitted for this examination — please indicate the make and model of your calculator on each answer book used.

1. (a) The `pthreads` library is a toolkit for writing parallel programs. What tools does it provide? [4 marks]

   (b) What is the principal difference between a *thread* and a *process*? [2 marks]

   (c) Explain the operation of the `pthread` library functions `pthread_mutex_lock` and `pthread_join`. [4 marks]

   (d) Imagine you have inputted a large colour picture, say a JPEG, and have expanded it into a two-dimensional array of pixels. Ten threads, each executing a copy of a thread function, cooperate to create a grey-level version of the picture in another array.

   - Write the thread function. [8 marks]
   - Write another function which creates the ten threads referred to above and which waits for them to complete before exiting. [2 marks]

   Notes:

   - A pixel's grey level can be calculated by averaging its red (R), green (G) and blue (B) intensity values.
   - *Do not write code to read in, decompress or to compress and write out the image* – marks will not be awarded for this.

2. (a) Tools such as SPIN and its associated modelling language Promela are radically different from most software development tools. Explain why this is so. [2 marks]

   (b) SPIN has a two principal modes of operation – interpretation and verification. Explain the difference between them, and explain its importance. [2 marks]

   (c) Explain the terms *deadlock*, *livelock* and *starvation*. [2 marks]

   (d) What is the Dining Philosophers Problem, and why is it of interest in the context of concurrent systems? [4 marks]

   (e) Write a Promela description of a reduced version of the Dining Philosophers Problem which has just two identical philosophers at a table for two and with just two forks. [4 marks]

   (f) Show how the system you describe will suffer from some combination of deadlock, livelock and/or starvation. [6 marks]

3. (a) What is the difference between a *virtual address* and a *physical address*? [2 marks]

   (b) Explain how the memory management part of a regular operating system implements virtual memory. [6 marks]

   (c) Virtual memory is normally not used in applications where assured real-time response is required. Why is that? [2 marks]

   (d) With reasonable values for access times to main memory (10 nanoseconds, i.e. $10 * 10^{-9}$ seconds) and backing store (10 milliseconds, i.e. $10 * 10^{-3}$ seconds), and with a page fault ratio of 1 in a million, calculate the average virtual memory access time of a system. [4 marks]

   (e) Explain, with diagrams, the operation of the *scheduler* in a conventional operating system. In your answer, explain the concept of *fairness* and explain how it might be managed in a scheduler. [6 marks]

Pthread Types and Function Prototypes

Definitions

```
pthread_t;  //this is the type of a pthread;
pthread_mutex_t; //this is the type of a mutex;
pthread_cond_t; // this is the type of a condition variable
```

Create a thread

```
int pthread_create(pthread_t *thread, pthread_attr_t *attr,
                   void *(*thread_function)(void *), void *arg);
```

Static Initialisation

Mutexes and condition variables can be initialized to default values using the `INITIALIZER` macros. For example:

```
pthread_mutex_t count_lock = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t count_cond = PTHREAD_COND_INITIALIZER;
```

Dynamic Initialisation

Mutexes, condition variables and semaphores can be initialized dynamically using the following calls:

```
int pthread_create(pthread_t *thread, pthread_attr_t *attr,
                   void *(*thread_function)(void *), void *arg);
int pthread_mutex_init(pthread_mutex_t *mutex,
                       const pthread_mutexattr_t *mutexattr);
int pthread_cond_init(pthread_cond_t *cond,
                      pthread_condattr_t *cond_attr);
int pthread_attr_init(pthread_attr_t *attr);
```

Deletion

```
int   pthread_mutex_destroy(pthread_mutex_t *);
int   pthread_cond_destroy(pthread_cond_t *);
```

Thread Function

The thread_function prototype would look like this:

```
void *thread_function(void *args);
```

Thread Exit & Join

```
void  pthread_exit(void *); // exit the thread i.e. terminate the thread
int   pthread_join(pthread_t, void **); // wait for the thread to exit.
```

Mutex locking and unlocking

```
int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_trylock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

Pthread Condition Variables

```
int pthread_cond_wait(pthread_cond_t *cond,
                      pthread_mutex_t *mutex);
int pthread_cond_signal(pthread_cond_t *cond);
int pthread_cond_broadcast(pthread_cond_t *cond);
```

Semaphores

```
sem_t; // this is the type of a semaphore
int sem_init(sem_t *sem, int pshared, unsigned int value); // pshared = 0 for semaphores
int sem_wait(sem_t *sp); // wait
int sem_post(sem_t *sp); // post
int sem_destroy(sem_t * sem); // delete
```