Lab Session 2:
Threads
21653-Sistemes Operatius

1 Introduction

The objective of this lab session is to program an application using threads, compare possible solutions and discuss the results. You will need to deliver two files:

- Anagrams.c.
- AnagramsThreaded.c.

You will also need to deliver a small document discussing your solutions and time benchmarking of the solutions. All together you will submit it in the aula global as a zip file. We have provided you with the files:

- US.txt: dictionary of English words
- anagram.c: computes if two words are anagrams.
- computetime.c: benchmarking time example

2 Computing Anagrams

An anagram of a word is another correct English word with the same number of letter occurrences. We have provided you with a text file containing all English words (US.txt).

You will need to compute all English anagrams of more than 5 letters. The program file must write in the standard output the anagrams as it founds them.

```
"silent-listen" are anagrams of 6 letters.
```

To read files you can use system calls or methods like "fopen" of stdio.h.

3 Computing Anagrams with Threads

You will need to compute anagrams using threads. You can for example dedicate one thread to search anagrams of a given word length. Think of different solutions and discuss them.

To create each thread it is not necessary that you use the attribute field. Here we remind you of the pthread methods:

```
#include <pthread.h>
pthread_t* tid;
pthread_create(&tid[i], NULL, codeThread, param);
pthread_join(tid[i], NULL);
```

Discuss the possibilities of loading all words into memory or accessing files from disk.

3.1 Synchronization

Threads place found anagrams in a central structure that you will need to access in a correct and synchronized way using a lock:

```
pthread_mutex_init(&lock, NULL)
pthread_mutex_lock(&lock);
pthread_mutex_unlock(&lock);
pthread_mutex_destroy(&lock);
```

The main thread writes all results in a file.