Operating Systems

Lab 09 Exercise – Shell scripts – Threads using Mutual Exclusion with `xchg`

Learning goals: this laboratory activity is devoted to the use of bash scripts and Mutual Exclusion with the busy form of waiting given by spin-lock.

Exercise 1

Write a C program that takes as arguments a number **`T`** and a directory name **`dir`**.

- The main thread, using the system call **`system`**, outputs in a file **`list.txt`** the list of files in directory **`dir`**, each including a random number of random integers. Then it reads the content of file **`list.txt`** in global memory, in an array of strings **`files`** with dimension given by the number of lines of file **`list.txt`**. Finally, it creates **`T`** threads and waits the all **`T`** threads have performed their sorting task before doing its own sorting task.
- Each created thread loops reading, in Mutual Exclusion, from **`files`** the next entry that it will sort by using system call **`system`**, until no more entries are available, then it displays its identity and the number of files that it has sorted, and exits.
- After the concurrent threads have sorted all files listed in **`list.txt`**, the main thread must produce a single file **`all_sorted.txt`**, where all the numbers appearing in all the sorted files are sorted in ascending order. Do this by using again system call **`system`** with the appropriate **`sort -m`** command.

You can protect a Critical Section by means of the access protocol:

```
acquire_lock;
CS ;
release_lock
```

where

```
int acquire_lock(int* lock){
int val = 1;
  while(val)
    __asm__("xchg %0, %1" : "+q" (val), "+m" (*lock));
  return 0;
}

int release_lock(int* lock){
  *lock = 0;
  return 0;
}
```

Exercise 2

Write a **bash** script that takes a filename argument, the file contains a number of file pathnames, which may correspond to regular, directories, links, or other special files.
The script must output the number of regular files that fulfill these conditions: the file is owned by the user executing the script, and have dimension greater than 1Kbyte.
If the pathname refers to a directory, the script must also print the directory pathname and take into account the regular files on that directory that fulfill the same conditions. No further processing is necessary for nested sub-directories.
Finally, the script must print the sum of the lines of all these files.

Exercise 3 is in the next page.

Exercise 3 exam

Write a **bash** script that takes as arguments a **`username`**, a group identifier (**`gid`**), and a **`string`** that represents the user first and last name. The script must append to file **`/etc/passwd`** a line that allows the system to recognize this user. In particular, the assigned user identifier (**`uid`**) will be computed by adding **`1`** to the **`uid`** of the user listed in the last row of file **`/etc/passwd`**, and the same interpreter.

This is an example of **`/etc/passwd`** line:

```
laface:x:1001:1001:Pietro Laface,,,:/home/laface:/bin/bash
```