Ejercicios

(a) Tenemos el siguiente programa fork_ping.py

```
import os
import sys

child_pid = os.fork()
if child_pid == 0:
        # child process
        os.system('ping -c 20 www.google.com > /ping.out')
        sys.exit(0)

print "En el padre, el pid del hijo es %d" % child_pid
pid, status = os.waitpid(child_pid, 0)
print "esperando volver, pid = %d, estado = %d" % (pid, status)
```

- Ejecútalo e intenta comprender qué es lo que realiza.
- Haz un programa, fork_ps.py, tomando como muestra este mismo, en el que el padre realice un ps -Af y exporte su salida a un fichero procesos.txt en el home de tu usuario.

  Mientras que el padre espera y, cuando ya ha realizado el hijo su tarea, cambia el nombre del archivo "procesos.txt" a "proc.txt".

(b) Este programa, base_thread.py, utiliza dos hilos para ejecución de la función "imprime".

```
import threading
import time

def imprime(cont, numhilo):
    while cont > 0:
        print "Hilo: %d Contando hacia atras %d " % (numhilo, cont)
        cont -= 1
        time.sleep(5)

t1 = threading.Thread(target=imprime, args=(5,1))
t2 = threading.Thread(target=imprime, args=(4,2))

t1.start(); t2.start()

t1.join(); t2.join()
```

- Ejecuta y comprende el programa.
- Modifica el código para que se ejecuten 4 hilos con contadores: 10, 4, 3, 7.

(c) En el siguiente programa, shared_value.py, se utiliza la variable x como global y hay dos hilos que la utilizan:

```
import time
import logging

x = 0           # x Compartido
def foo():
    global x
    for i in xrange(100000000): x += 1

def bar():
    global x
    for i in xrange(100000000): x -= 1

t1 = threading.Thread(target=foo)
t2 = threading.Thread(target=bar)
t1.start(); t2.start()
t1.join(); t2.join()   # Espera a que se completen
print x                # El resultado esperado es 0
```

- ¿Cuál es la salida? ¿Por qué?
- Modifica el programa para que el resultado sea 0.