Actividad 7: Ejercicios sobre comunicación entre procesos

Dra. Magali Arellano Vázquez

Instrucciones generales:

- En los ejercicios que en que se soliciten **códigos**, es necesario incluir, además del código correspondiente, las capturas de pantalla que sustente su correcto funcionamiento con la configuración solicitada (con una máquina, dos, etc).
- Si el ejercicio solicita **diseño de un algoritmo o estrategia**, este diseño debe ser argumentado,
- En los ejercicios que se soliciten **investigación**, evita hacer copias textuales del material consultado, tampoco olvides poner las referencias bibliográficas correspondientes.

1. **Reporte**
   Analice la ejecución de cada uno de estos códigos.

   - Elabore un reporte y diga cual es la diferencia entre ejecutar esto con hilos y con ciclos for u otra estructura de control ¿qué utilidad le ve a los hilos?

Código 1 El siguiente código crea dos hilos, cada uno de los cuales realiza 10 iteraciones, en cada una de las cuales muestra por pantalla su nombre y el número de la iteración.

- Copie y ejecute el siguiente código:

```
1  import time
2  from threading import Thread
3
4  class MiHilo( Thread ):
5
6          def __init__( self,  num, nom ):
7
8                  Thread.__init__(self)
9                  self.bignum = num
10                 self.nombre = nom
11
12         def run( self ):
13
14                 for l in range( 10 ):
15                         for k in range( self.bignum ):
16                                 res = 0
17                                 for i in range( self.bignum ):
18                                         res += 1
19                         print( self.nombre + ": iteracion " + str( l ) )
20 def test():
21         num1 = 1000
22         num2 = 500
23         thr1 = MiHilo( num1, "Hilo 1" )
24         thr2 = MiHilo( num2, "Hilo 2" )
25         thr2.start()
26         thr1.start()
27         thr1.join()
28         thr2.join()
29
30 if __name__ == "__main__":
31         test()
```

Código 2 El siguiente código hace `ping` a un conjunto de host (cambiar el host por uno que esté dentro de una subred)

dentro de una red y determina su estatus.

- Copie y ejecute el siguiente código:

```
1
2  # -*- coding: utf-8 -*-
3  import os
4  import re
5  import time
6  import sys
7  from threading import Thread
8
9  class testit(Thread):
10     def __init__ (self,ip):
11         Thread.__init__(self)
12         self.ip = ip
13         self.status = -1
14
15     def run(self):
16         pingaling = os.popen("ping -q -c2 "+self.ip,"r")
17         while 1:
18           line = pingaling.readline()
19           if not line: break
20           igot = re.findall(testit.lifeline,line)
21           if igot:
22               self.status = int(igot[0])
23
24
25 testit.lifeline = re.compile(r"(\d) received")
26 report = ("No response","Partial Response","Alive")
27
28 print (time.ctime())
29
30 pinglist = [ ]
31
32 for host in range(40,50):
33     ip = "192.168.135."+str(host)
34     current = testit(ip)
35     pinglist.append(current)
36     current.start()
37
38 for pingle in pinglist:
39     pingle.join()
40     print ("Status from ",pingle.ip,"is",report[pingle.status])
41
42 print (time.ctime())
```

2. **Hilos literarios**
   Cuando en un programa tenemos varios hilos en ejecución

simultanea es posible que varios de ellos intenten acceder a la región crítica (un fichero, una conexión, un array) y que la operación de uno de ellos entorpezca la de los otros. Para evitar estos problemas, es necesaria la sincronización. Por ejemplo, si un hilo con vocación de Cervantes escribe en el fichero “El Quijote” y el otro con vocación de Shakespeare escribe “Romeo y Julieta”, al final quedarán todas las letras entremezcladas. Se busca que uno escriba primero su “Quijote” y luego el otro su “Romeo y Julieta”.

El objetivo es crear 3 hilos que serán: Quijote, Romeo y Julieta; escribiendo en el mismo fichero y ejecutándose simultáneamente.

El hilo Quijote escribirá:

"...En un lugar de la Mancha de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.
Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los viernes..."

El hilo “Romeo” escribirá:

- Habla. ¡Oh! ¡Habla otra vez ángel resplandeciente!... Porque esta noche apareces tan esplendorosa sobre mi cabeza como un alado mensajero celeste ante los ojos estáticos y maravillados de los mortales, que se inclinan hacia atrás para verle, cuando él cabalga sobre las tardas perezosas nubes y navega en el seno del aire.

Y el hilo Julieta escribirá:

¡Oh Romeo, Romeo! ¿Por qué eres tú Romeo? Niega a tu padre y rehusa tu nombre; o, si no quieres, júrame tan sólo que me amas, y dejaré yo de ser una Capuleto.

En el archivo deberá escribirse en ese orden (Quijote-Romeo-Julieta), utilizando los mecanismos de sincronización de los hilos.

3. Diseñar y codificar un programa que resuelva el siguiente problema, puede ser en cualquier lenguaje que soporte el uso de hilos (el ejemplo se da en python).

   Cuenta palabras Realizar un programa que cuente las palabras de los archivos de texto que se le pasen como parámetros.

   El programa debe imprimir el número de palabras totales (contando las de todos los ficheros juntos). Por ejemplo: ./python cuenta.py file1 file2 file3 debería generar una salida como la siguiente (se asume que file1 tiene 100 palabras, file2 200 y file3 300):

```
$ ./python cuenta.py file1 file2 file3
file1: 100 palabras
file2: 200 palabras
file3: 300 palabras
total: 600 palabras
$
```

   El programa debe crear un hilo por cada archivo para permitir que las palabras de todos los ficheros se cuenten concurrentemente. Deben asegurarse que al final los ficheros aparezcan listados en el mismo orden en el que fueron especificados en la linea de comandos.

   Nota: Ten en cuenta que el programa deberá recibir los nombres de los archivos como entrada, en el ejemplo se muestra con 3 archivos pero el programa debería ser capaz de recibir cualquier número de archivos.

4. Se les proporcionarán 2 códigos:

```
cliente.py
server.py
```

   Al ejecutar los códigos proporcionados, al ejecutar primero el servidor, este abre el socket, el socket estará esperando una petición de un cliente y mandará un mensaje. Al ejecutar el

cliente se muestra el mensaje: `Hola Mundo!` Utilizando como base estos códigos, modificarlos de manera que funcionen como un chat.

5. Se les proporcionarán 2 códigos:

   ```
   rpccliente.py
   rpcserver.py
   ```

   En estos códigos está implementada la función suma, tomando como ejemplo esta, agrega otras 5 funcionalidades al código. Estos códigos son solo la base para crear programas mas complejos en los que intervienen funciones mas avanzadas y especificas de la implementación de XML-RPC de python. Para esta practica requieren tener 2 equipos conectados a la misma subred, el número de puerto lo pueden cambiar (recuerda poner un valor alto en el número de puerto, para evitar colisionar con los puertos predefinidos del sistema). El servidor y el cliente deben ejecutarse en distintos equipos.

6. Diseña un algoritmo (puede ser solo pseudocódigo) que solucione el problema de los filósofos comensales, que se encuentra en la página 9 de la presentación Comunicación entre procesos. Considera que no deben presentarse las condiciones de starvation ni deadlock. Recuerda argumentar tu solución, sobre todo las ventajas y desventajas.

7. Los algoritmos de exclusión mutua (comúnmente abreviada como mutex por mutual exclusion) se usan en programación concurrente para evitar que entre más de un proceso a la vez en la sección crítica. Realiza una investigación sobre :

   - El algoritmo de Dekker.
   - La solución de Peterson.