PROGRAMACIÓN DE SERVICIOS Y PROCESOS

Unidad 1: Procesos e Hilos

Actividad. Sincronización de Procesos con Waitfor

En programación multiproceso, varios procesos se turnan la CPU para avanzar de forma aparentemente simultánea para el usuario. El sistema operativo cede un espacio de memoria independiente para cada proceso y se encarga de su gestión.

Si los procesos cooperan entre ellos para obtener un resultado, es trabajo del programador establecer esa comunicación y sincronización

A) Programa java que acepte como argumento un comando o programa y lo ejecute como un proceso hijo. Vuestro programa esperará a que finalice y mostrará el valor de salida del proceso hijo, que podrá ser 0 o 1. Capturamos las excepciones informando del error. Si se ejecuta sin argumentos, el programa mostrará un mensaje que indique que falta el comando

Nuestro programa está escrito de forma que, al no introducir ningún argumento por parámetro al ejecutar el archivo .jar, nos muestra un mensaje avisándonos de que no hemos introducido ningún parámetro y por lo tanto no se realizará ninguna acción.

```
PS C:\Users\dam2\Documents\eclipse-workspace\DAM2\Sincronizacion_AntonioBenitez> java -jar Sincronizacion.jar
No se ha introducido ningún argumento.
```

B) Exporta el proyecto a .jar ejecutable. Probar la ejecución desde un terminal (CMD) usando estos argumentos, y explicando la salida que obtenemos:

- **ping de 1 datagrama a un host que devuelva echo**
- **ping de 1 datagrama a un host que no devuelva echo**
- **El block de notas**
- **un programa python donde se retorne 0 o 1 de forma aleatoria**

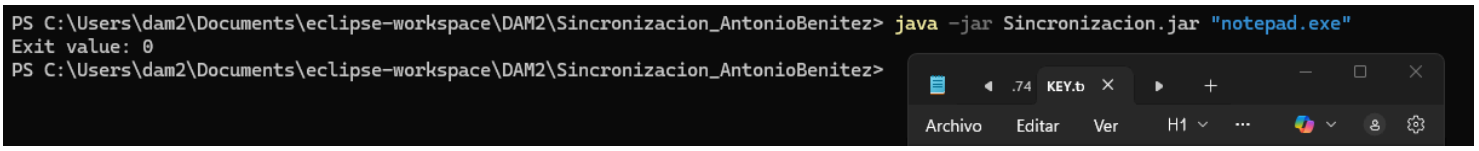
Para empezar, vamos a ejecutar nuestro programa JAR introduciendo la instrucción “ping localhost” (un ping a nuestra propia IP que siempre dará respuesta) por parámetro. Cuando ejecutamos esta orden mediante nuestro programa JAR, nos devuelve un Exit Value de 0.

```
PS C:\Users\dam2\Documents\eclipse-workspace\DAM2\Sincronizacion_AntonioBenitez> java -jar Sincronizacion.jar ping localhost
Exit value: 0
```

A continuación, vamos a introducir por parámetro la instrucción “ping 192.168.5.5” (un ping erróneo a una IP no existente en la red). Al ejecutar el programa JAR con esta orden, nos devuelve un Exit Value de 1, pues el programa de ping ha sufrido un error debido a que la IP que hemos introducido no existe.

```
PS C:\Users\dam2\Documents\eclipse-workspace\DAM2\Sincronizacion_AntonioBenitez> java -jar Sincronizacion.jar ping 192.168.5.5
Exit value: 1
```

Para el siguiente parámetro, vamos a ejecutar la orden “notepad.exe” mediante nuestro programa JAR. Al ejecutarlo, nos abre exitosamente el programa de Bloc de Notas y nos devuelve un Exit Code de 0, pues se ha ejecutado correctamente.



Por último, vamos a introducir por parámetro la instrucción “python aleatorio.py”; es decir, que nuestro programa JAR va a ejecutar un programa Python tras haber introducido la orden por parámetro. Al ejecutarlo, podemos ver que ejecuta el programa Python de forma correcta, pues nos devuelve un valor aleatorio entre 0 y 1 como su Exit Value.

```
PS C:\Users\dam2\Documents\eclipse-workspace\DAM2\Sincronizacion_AntonioBenitez> java -jar Sincronizacion.jar python aleatorio.py
Exit value: 1
PS C:\Users\dam2\Documents\eclipse-workspace\DAM2\Sincronizacion_AntonioBenitez> java -jar Sincronizacion.jar python aleatorio.py
Exit value: 1
PS C:\Users\dam2\Documents\eclipse-workspace\DAM2\Sincronizacion_AntonioBenitez> java -jar Sincronizacion.jar python aleatorio.py
Exit value: 0
PS C:\Users\dam2\Documents\eclipse-workspace\DAM2\Sincronizacion_AntonioBenitez>
```