|  |
| --- |
| Tema 1 |
|  |
| Programación de servicios y procesos 2ºDAM  Rodrigo Tapiador Cano |

[1.1. Sistema Operativo Multitarea 3](#_Toc190171139)

[1.2. Lenguajes de Programación Compilados 3](#_Toc190171140)

[1.3. Identificador de Proceso 3](#_Toc190171141)

[1.4. Procesamiento Distribuido 4](#_Toc190171142)

[1.5. Planificador de Procesos del Sistema Operativo 4](#_Toc190171143)

[1.6. Estado de un Proceso 4](#_Toc190171144)

[1.7. Clase Runtime de Java 4](#_Toc190171145)

[1.8. Método de la Clase Process de Java 5](#_Toc190171146)

[2. Actividades de Aplicación 5](#_Toc190171147)

[2.1. Diferencias entre Lenguajes de Programación 5](#_Toc190171148)

[2.2. Programa vs Proceso 5](#_Toc190171149)

[2.3. Programación Distribuida 5](#_Toc190171150)

[2.4. Ejecutar Procesos vs Bifurcación 6](#_Toc190171151)

[2.5. Intercambio de Información 6](#_Toc190171152)

[2.6. Sincronización de Procesos 8](#_Toc190171153)

[2.7. Sincronización en Ordenadores Independientes 9](#_Toc190171154)

[2.8. Consumo de Recursos 9](#_Toc190171155)

[2.9. Planificador de Procesos del Sistema Operativo 10](#_Toc190171156)

[2.10. Diseño de un Planificador de Procesos 10](#_Toc190171157)

[2.11. Listado de Valores y Descripciones 11](#_Toc190171158)

[2.12. Programa en Java 12](#_Toc190171159)

## Sistema Operativo Multitarea

### ¿Cuál de los siguientes sistemas operativos no es multitarea?

Respuesta: d) MS-DOS.

Justificación: El sistema operativo MS-DOS no soporta la multitarea, es decir, no puede ejecutar múltiples programas al mismo tiempo. Solo permite la ejecución de un programa a la vez.

Referencia: [Ms-DOS](https://es.wikipedia.org/wiki/MS-DOS)

### En computación, se entiende por multitarea:

Respuesta: d) La capacidad que tiene un ordenador de ejecutar varios programas al mismo tiempo.

Justificación: La multitarea permite que varios programas se ejecuten simultáneamente, optimizando el uso de recursos del sistema operativo.

Referencia: [Multitarea](https://es.wikipedia.org/wiki/Multitarea)

### ¿Qué elemento del ordenador se encarga de gestionar la multitarea?

Respuesta: b) El sistema operativo.

Justificación: Es el sistema operativo quien se encarga de la gestión de la multitarea, administrando los recursos del ordenador para permitir la ejecución simultánea de múltiples programas.

Referencia: [Multitarea](https://es.wikipedia.org/wiki/Multitarea)

## Lenguajes de Programación Compilados

### ¿Qué afirmación referente a los lenguajes de programación compilados no es correcta?

Respuesta: c) Si el lenguaje es estándar, la misma compilación sirve para todos los sistemas operativos.

Justificación: Los lenguajes de programación compilados necesitan ser recompilados para cada sistema operativo específico, debido a las diferencias en las arquitecturas y en las API del sistema.

Referencia: [Compilador](https://es.wikipedia.org/wiki/Compilador)

## Identificador de Proceso

### El identificador de proceso se suele identificar por las siglas:

Respuesta: b) PID.

Justificación: PID significa "Process Identifier" y es un número único asignado por el sistema operativo para identificar cada proceso.

Referencia: [PID](https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/finding-the-process-id)

## Procesamiento Distribuido

### El procesamiento que se ejecuta en diferentes ordenadores independientes, pero conectados y sincronizados se denomina:

Respuesta: a) Distribuido.

Justificación: La computación distribuida se refiere a sistemas donde múltiples ordenadores independientes trabajan en conjunto para completar una tarea.

Referencia: [Computación distribuida](https://es.wikipedia.org/wiki/Computaci%C3%B3n_distribuida)

## Planificador de Procesos del Sistema Operativo

### ¿Cuál de los siguientes no es un objetivo del planificador de procesos del sistema operativo?

Respuesta: b) Maximizar los tiempos de respuesta.

Justificación: El objetivo del planificador de procesos es minimizar los tiempos de respuesta, maximizar el rendimiento del sistema y repartir los recursos de manera equitativa.

Referencia: [Planificador de procesos](https://ocw.uc3m.es/pluginfile.php/3335/mod_page/content/19/planificacion_procesos.pdf)

## Estado de un Proceso

### ¿Cuál es el estado al que puede pasar un proceso que está en estado Listo?

Respuesta: c) En ejecución.

Justificación: Un proceso en estado listo puede ser seleccionado por el planificador de procesos para pasar a estado de ejecución.

Referencia: [Estados de los procesos](https://www.profesionalreview.com/2020/06/25/cuales-son-los-estados-de-los-procesos-de-nuestros-equipos/)

## Clase Runtime de Java

### ¿Qué afirmación referente a la clase Runtime de Java es errónea?

Respuesta: c) Permite conocer el valor de las variables internas del proceso lanzado.

Justificación: La clase Runtime en Java no permite conocer el valor de las variables internas del proceso lanzado. Permite ejecutar un proceso, esperar su finalización y obtener su estado de terminación.

Referencia: [Runtime](https://docs.oracle.com/javase/8/docs/api/java/lang/Runtime.html)

## Método de la Clase Process de Java

### ¿Cómo se llama el método de la clase Process de Java que hace esperar a que termine el proceso lanzado?

Respuesta: c) waitFor().

Justificación: El método waitFor() de la clase Process en Java hace que el proceso actual espere a que termine el proceso lanzado.

Referencia: [Process](https://docs.oracle.com/javase/8/docs/api/java/lang/Process.html#waitFor--)

# Actividades de Aplicación

## Diferencias entre Lenguajes de Programación

### Explica las diferencias entre los lenguajes de programación interpretados y compilados.

Los lenguajes compilados son transformados directamente a código de máquina ejecutable, lo que suele resultar en un mejor rendimiento. Por otro lado, los lenguajes interpretados son traducidos y ejecutados línea por línea en tiempo de ejecución, lo que generalmente conlleva una ejecución más lenta, pero permite una mayor flexibilidad y facilidad para la depuración.

Referencia: [Lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n)

## Programa vs Proceso

### Describe las diferencias entre programa y proceso.

Un programa es un conjunto de instrucciones que están escritas en un lenguaje de programación. Un proceso es la ejecución de esas instrucciones por el sistema operativo. Mientras que un programa es una entidad estática, un proceso es dinámico y tiene estado (en ejecución, bloqueado, listo).

Referencia: [Proceso](https://es.wikipedia.org/wiki/Proceso_(inform%C3%A1tica))

## Programación Distribuida

### Explica en qué consiste la programación distribuida.

La programación distribuida se refiere al desarrollo de software para sistemas distribuidos, en los cuales varios ordenadores independientes se comunican y colaboran para realizar tareas. Esto permite mejorar la eficiencia, escalabilidad y tolerancia a fallos.

Referencia: [Computación distribuida](https://es.wikipedia.org/wiki/Computaci%C3%B3n_distribuida)

## Ejecutar Procesos vs Bifurcación

### Encuentra las diferencias entre ejecutar dos procesos o realizar una bifurcación o fork.

Ejecutar dos procesos puede hacerse iniciando dos instancias de un programa independientemente. Realizar una bifurcación, o fork, implica crear un nuevo proceso duplicado del proceso que llama a fork, compartiendo el mismo espacio de memoria.

Referencia: [fork](https://es.wikipedia.org/wiki/Bifurcaci%C3%B3n_(desarrollo_de_software))

## Intercambio de Información

### Diseña un sistema basado en bases de datos para intercambiar información entre dos procesos.

Un sistema basado en bases de datos podría utilizar una tabla compartida a la que ambos procesos tienen acceso. Los procesos pueden escribir y leer datos en esta tabla, utilizando técnicas como bloqueo y transacciones para evitar conflictos y asegurar la consistencia de los datos.

Conexión a base de datos:

import sqlite3

import time

import multiprocessing

*class* ProcessStateDB:

*def* \_\_init\_\_(*self*, *db\_name*):

*self*.db\_name = db\_name

*self*.conn = sqlite3.connect(*self*.db\_name)

*self*.create\_table()

*def* create\_table(*self*):

        with *self*.conn:

*self*.conn.execute('''CREATE TABLE IF NOT EXISTS process\_state (

                                    process\_id INTEGER PRIMARY KEY,

                                    state TEXT

                                 );''')

*def* write\_state(*self*, *process\_id*, *state*):

        with *self*.conn:

*self*.conn.execute('REPLACE INTO process\_state (process\_id, state) VALUES (?, ?);', (process\_id, state))

*def* read\_state(*self*, *process\_id*):

        cursor = *self*.conn.cursor()

        cursor.execute('SELECT state FROM process\_state WHERE process\_id = ?;', (process\_id,))

        result = cursor.fetchone()

        return result[0] if result else None

*def* close(*self*):

*self*.conn.close()

Función que ejecutará el hilo:

*def* process\_function(*process\_id*, *db\_name*):

    process\_state\_db = ProcessStateDB(db\_name)

    states = ["INICIO", "PROCESANDO", "FINALIZADO"]

    for state in states:

        time.sleep(1)

        print(*f*"Proceso {process\_id} cambiando estado a: {state}")

        process\_state\_db.write\_state(process\_id, state)

    process\_state\_db.close()

Main:

if \_\_name\_\_ == "\_\_main\_\_":

    db\_name = "process\_state.db"

    process\_state\_db = ProcessStateDB(db\_name)

    processes = []

    for i in range(2):

        p = multiprocessing.Process(*target*=process\_function, *args*=(i, db\_name))

        processes.append(p)

        p.start()

    for p in processes:

        p.join()

    for i in range(2):

        print(*f*"Estado final del proceso {i}: {process\_state\_db.read\_state(i)}")

    process\_state\_db.close()

Referencia: [sqlite3](https://docs.python.org/3/library/sqlite3.html)

## Sincronización de Procesos

### Diseña un sistema basado en ficheros de texto para sincronizar procesos.

En este sistema, los procesos pueden escribir y leer en archivos de texto compartidos. Se pueden utilizar mecanismos de bloqueo de archivos para asegurar que un solo proceso pueda modificar el archivo en un momento dado, evitando inconsistencias.

Creamos una clase que permita escribir en fichero, bloqueándolo, donde escribiremos el estado de los procesos para que el otro proceso pueda leerlo

import os

import time

import portalocker as pl

*class* ProcessState:

*def* \_\_init\_\_(*self*, *filename*):

*self*.filename = filename

*def* lock\_file(*self*, *mode*):

*self*.file = open(*self*.filename, mode)

        pl.lock(*self*.file, pl.LOCK\_EX)

*def* unlock\_file(*self*):

        pl.lock(*self*.file, pl.LOCK\_UN)

*self*.file.close()

*def* write\_state(*self*, *state*):

*self*.lock\_file('w')

*self*.file.write(state)

*self*.unlock\_file()

*def* read\_state(*self*):

*self*.lock\_file('r')

        state = *self*.file.read().strip()

*self*.unlock\_file()

        return state

Hacemos una función que podamos ejecutar como un hilo:

import multiprocessing

*def* process\_function(*process\_id*, *process\_state*):

*#simula diferentes estados del proceso*

    states = ["INICIO", "PROCESANDO", "FINALIZADO"]

    for state in states:

        time.sleep(1)  *#simular tiempo de procesamiento*

        print(*f*"Proceso {process\_id} cambiando estado a: {state}")

        process\_state.write\_state(state) *#Escribe el estado en el que se encuentra el proceso*

Creamos un main donde probar el código:

if \_\_name\_\_ == "\_\_main\_\_":

    filename = "process\_state.txt"

*#asegurarse de que el archivo exista*

    if not os.path.exists(filename):

        open(filename, 'w').close()

    process\_state = ProcessState(filename) *#Se crea un objeto process\_state*

    processes = []

    for i in range(2):

        p = multiprocessing.Process(*target*=process\_function, *args*=(i, process\_state)) *#Se crea un hilo de la clase Process que ejecuta la función process\_function con los argumentos necesarios*

        processes.append(p)

        p.start()

    for p in processes:

        p.join() *#Esperamos a que terminen los procesos*

*# Mostrar el estado final del fichero*

    print(*f*"Estado final del fichero: {process\_state.read\_state()}")

Referencia: [Portalocker](https://stackoverflow.com/questions/45228395/error-no-module-named-fcntl)

## Sincronización en Ordenadores Independientes

### Decide qué sistema de sincronización podría servir para sincronizar procesos que se ejecutan en ordenadores independientes conectados a través de internet.

Para sincronizar procesos en ordenadores independientes a través de internet, se podría utilizar un sistema de mensajería, que permita la comunicación y sincronización de procesos a través de mensajes.

Referencia: [Paso de mensajes](https://www.linkedin.com/advice/1/how-can-you-use-message-passing-synchronize-processes?lang=es&originalSubdomain=es#:~:text=El%20paso%20de%20mensajes%20es%20un%20m%C3%A9todo%20de,una%20canalizaci%C3%B3n%2C%20un%20socket%20o%20una%20memoria%20compartida.)

## Consumo de Recursos

### Explica por qué los hilos (threads) consumen menos recursos que los procesos.

Los hilos comparten el mismo espacio de memoria y recursos de un proceso, por lo que crean menos sobrecarga en términos de memoria y tiempo de creación/destrucción comparada con procesos, que tienen su propio espacio de memoria y recursos.

Referencia: [Proceso vs hilo](https://www.guru99.com/es/difference-between-process-and-thread.html#:~:text=Los%20subprocesos%20consumen%20menos%20recursos.%20El%20sistema%20operativo,comparten%20memoria.%20Los%20hilos%20comparten%20datos%20entre%20s%C3%AD.)

## Planificador de Procesos del Sistema Operativo

### Explica qué tarea cumple el planificador de procesos dentro del sistema operativo.

El planificador de procesos administra la CPU, asignando tiempos de ejecución a los diferentes procesos para asegurar que todos puedan avanzar de manera justa y eficiente. Su objetivo es maximizar el uso de la CPU, minimizar tiempos de espera y equilibrar la carga de trabajo.

Referencia: [Planificación de procesos](https://sistop.gwolf.org/html/03_planificacion_de_procesos.html)

## Diseño de un Planificador de Procesos

### Diseña un planificador de proceso sencillo. Determina cómo se asigna el tiempo de CPU a los distintos procesos de manera que se puedan avanzar todos más o menos simultáneamente.

Si hacemos una planificación mediante round robin tendríamos:

Clase Process con tiempo esperado de ejecución y método run for quantum:

import threading

import time

*# Definir el quantum (tiempo de CPU asignado a cada proceso)*

QUANTUM = 2  *# segundos*

*class* Process(*threading*.*Thread*):

*def* \_\_init\_\_(*self*, *process\_id*, *burst\_time*):

        threading.Thread.\_\_init\_\_(*self*)

*self*.process\_id = process\_id

*self*.burst\_time = burst\_time

*def* run\_for\_quantum(*self*):

        if *self*.burst\_time > 0:

            time\_to\_run = min(*self*.burst\_time, QUANTUM)

            print(*f*"Proceso {*self*.process\_id} ejecutándose por {time\_to\_run} segundos.")

            time.sleep(time\_to\_run)

*self*.burst\_time -= time\_to\_run

            if *self*.burst\_time > 0:

                print(*f*"Proceso {*self*.process\_id} detenido, resta {*self*.burst\_time} segundos.")

            else:

                print(*f*"Proceso {*self*.process\_id} finalizado.")

Función scheduler que organiza los procesos:

*def* round\_robin\_scheduler(*processes*):

    while any(process.burst\_time > 0 for process in processes):

        for process in processes:

            if process.burst\_time > 0:

                process.run\_for\_quantum()

Main:

if \_\_name\_\_ == "\_\_main\_\_":

*# Crear procesos con diferentes tiempos de ráfaga (burst time)*

    processes = [

        Process(1, 5),

        Process(2, 8),

        Process(3, 3),

        Process(4, 6)

    ]

*# Iniciar el planificador Round Robin*

    round\_robin\_scheduler(processes)

## Listado de Valores y Descripciones

### Elabora el listado de los valores y sus descripciones que podría devolver la ejecución de un proceso que tenga que acceder a una base de datos, a un ordenador a través de la red y a internet.

1. **SUCCESS**: La operación se realizó exitosamente.

**Descripción**: El proceso logró completar la operación sin errores.

1. **DATABASE\_CONNECTION\_FAILED**: Fallo en la conexión a la base de datos.

**Descripción**: No se pudo establecer una conexión con la base de datos. Esto podría deberse a credenciales incorrectas, problemas de red, o que la base de datos no esté disponible.

1. **NETWORK\_TIMEOUT**: Tiempo de espera de red agotado.

**Descripción**: La operación de red tomó más tiempo del esperado y se agotó el tiempo de espera configurado.

1. **NTERNET\_DISCONNECTED**: Desconexión de Internet.

**Descripción**: El proceso no pudo acceder a Internet debido a una desconexión.

1. **DATABASE\_QUERY\_FAILED**: Error en la ejecución de la consulta de la base de datos.

**Descripción**: La consulta SQL no pudo ejecutarse correctamente. Esto podría deberse a errores sintácticos, problemas de permisos, o inconsistencias en los datos.

1. **RESOURCE\_NOT\_FOUND**: Recurso no encontrado.

**Descripción**: El recurso solicitado (como un archivo en el ordenador remoto o un dato en la base de datos) no se encontró.

1. **PERMISSION\_DENIED**: Permiso denegado.

**Descripción**: El proceso no tiene los permisos necesarios para realizar la operación solicitada.

1. I**NVALID\_INPUT**: Entrada inválida.

**Descripción**: Los datos proporcionados al proceso no son válidos o no cumplen con los requisitos necesarios para la operación.

1. **AUTHENTICATION\_FAILED**: Fallo en la autenticación.

**Descripción**: No se pudo autenticar correctamente con el servicio remoto, la base de datos o el recurso de red. Esto podría deberse a credenciales incorrectas o políticas de seguridad.

1. **UNKNOWN\_ERROR**: Error desconocido.

**Descripción**: Ocurrió un error que no se pudo clasificar en ninguna de las categorías anteriores.

1. **SERVICE\_UNAVAILABLE**: Servicio no disponible.

**Descripción**: El servicio al que se intentaba acceder está temporalmente fuera de servicio.

1. **DATA\_INTEGRITY\_ERROR**: Error de integridad de los datos.

**Descripción**: Se detectó un problema de integridad en los datos durante la operación.

1. **DISK\_SPACE\_EXCEEDED**: Espacio en disco excedido.

**Descripción**: No hay suficiente espacio en disco para completar la operación.

1. **RATE\_LIMIT\_EXCEEDED**: Límite de tasa excedido.

**Descripción**: Se ha excedido el límite de solicitudes permitidas en un período de tiempo determinado.

Referencias: [Exception handling](https://en.wikipedia.org/wiki/Exception_handling) [http error codes](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status)

## Programa en Java

### Desarrolla un programa en Java que ejecute el navegador web Firefox. Utiliza la clase Runtime. Indica que el navegador abra directamente una página web.

public *class* Main {

    public static void main(String[] args) {

        try {

            // URL de la página web que deseas abrir

            String url = "https://www.example.com";

            // Comando para abrir Microsoft Edge con la URL especificada

            String command = "cmd /c start msedge " + url;

            // Ejecutar el comando

            Runtime.getRuntime().exec(command);

            System.out.println("Navegador Edge abierto con la URL: " + url);

        } catch (*Exception* e) {

            e.printStackTrace();

        }

    }

}