



UNIVERSIDAD  
AUTÓNOMA  
METROPOLITANA  
**Unidad Cuajimalpa**

# Tarea 4

## Sistemas Operativos

Jorge Angel Juárez Vázquez  
2213026247

Profesor: Jose Netz Romero Duran  
14 September 2024

# Tarea 4 - Memoria compartida

## Nota

Negocio: WoldFork

Lenguaje de programación: Python 

## Archivo

Para la primera fase de la tarea se hizo uso de un archivo `.csv` que funcione como tipo base de datos, a los registros de los productos, se le agregó nuevo campo el cual será el de `stock` que representa la cantidad de artículos en existencia, el contenido del archivo se ve de la siguiente forma:

```
id_reg,name,category,weight,price,expiry,stock
1,Nuggets de pollo,Aves,0.8,120,2024-10-30,27
2,Pan de ajo grande,Panaderia,0.3,50,2024-10-15,29
3,Yogurt natural nuez,Lacteos,0.5,25,2024-10-30,40
4,Filete de res Angus,Carne,1.2,250,2024-12-31,20
5,Helado de vainilla,Helados,1.5,100,2024-09-30,10
6,Bolsa de vegetales,Vegetales,0.5,80,2024-12-10,12
7,Galletas de nuez,Galletas,0.2,20,2024-11-15,21
8,Camarones jumbo,Mariscos,0.5,180,2024-11-15,5
9,Salsa de tomate,Salsas,0.5,30,2024-12-31,12
10,Pan de ajo chico,Panaderia,0.1,20,2024-10-15,30
```

El cual va a ser leído y cada registro va a ser representado por un objeto de tipo `Register`, para que sea más fácil la manipulación de los mismos en las siguientes fases del desarrollo, aparte de la clase para representar cada registro, se crearon funciones para la lectura del archivo `.csv`, escritura en el archivo `.csv`, para crear una fila de una tabla con la información del registro (tabla mostrada en la interfaz del sistema), y para buscar un registro en una lista de objetos de tipo `Register`; esta clase y funciones fueron codificadas de la siguiente forma:

`core.py`

```
from typing import List, Dict, Union, Optional
from csv import DictReader, DictWriter
import flet as ft

class Register:
    def __init__(self, id_reg: int, name: str, category: str, weight: int,
                 price: float, expiry: str, stock: int, is_locked: Optional[bool] = None) →
```

None:

```
    self.id_reg = id_reg
    self.name = name
    self.category = category
    self.weight = weight
    self.price = price
    self.expiry = expiry
    self.stock = stock
    self.is_locked = is_locked

def getattribs_dict(self) → Dict:
    return {
        "id_reg": self.id_reg, "name": self.name, "category": self.category,
        "weight": self.weight, "price": self.price,
        "expiry": self.expiry, "stock": self.stock
    }

def __str__(self) → str:
    return f'Nombre: {self.name}\nCategoria: {self.category}\n\nPeso (kg): {self.weight}\nPrecio (MXN): $ {self.price}\n\nCaducidad: {self.expiry}\nStock: {self.stock}'

def read_csv(file_path: str) → List[Register]:
    """Leer datos desde un archivo CSV y devolver una lista de registros."""
    try:
        with open(file_path, mode='r', encoding='utf-8') as file:
            csv_reader = DictReader(file)
            return [Register(**row) for row in csv_reader]
    except FileNotFoundError:
        print(f"El archivo {file_path} no se encuentra.")
        return []

def write_csv(file_path: str, registers: List[Register]) → None:
    """Escribir los registros actualizados en un archivo CSV."""
    with open(file_path, mode='w', newline='', encoding='utf-8') as csvfile:
        fieldnames = ['id_reg', 'name', 'category', 'weight',
                      'price', 'expiry', 'stock']

        writer = DictWriter(csvfile, fieldnames=fieldnames)
        writer.writeheader()
        for r in registers:
            writer.writerow(r.getattribs_dict())

def to_DataTable(registers: List[Register]) → List[ft.DataRow]:
    """Convertir los registros en filas para la tabla de datos en Flet."""
    return [
        ftDataRow(
```

```

        cells=[
            ft.DataCell(ft.Text(str(rr.id_reg))),
            ft.DataCell(ft.Text(rr.name)),
            ft.DataCell(ft.Text(rr.category)),
            ft.DataCell(ft.Text(str(rr.weight))),
            ft.DataCell(ft.Text(f"${rr.price}")),
            ft.DataCell(ft.Text(rr.expiry)),
            ft.DataCell(ft.Text(str(rr.stock)))
        ]
    ) for rr in registers
]

def search_register(registers: List[Register], id_reg: int) → Union[Register, bool]:
    """Buscar un registro por su ID. Devolver el registro o False si no existe."""
    return False if id_reg > len(registers) or id_reg ≤ 0 else
    registers[id_reg - 1]

```

## Interfaz gráfica

Para la creación de la interfaz gráfica del sistema se ocupó **felt** que es un framework que permite crear aplicaciones web utilizando Python, sin necesidad de escribir código HTML, CSS o JavaScript. Utiliza un modelo similar a Flutter (de Google), donde puedes crear interfaces de usuario mediante la programación de componentes en Python. Es útil para construir aplicaciones interactivas y modernas, con soporte para widgets, notificaciones y funcionalidades más avanzadas de forma sencilla.

El código que se desarrolló fue el siguiente:

app.py

```

import flet as ft
from core import read_csv, write_csv, to_DataTable, search_register

# Ruta del archivo CSV
DATA_FILE = "./data/wildfork.csv"

# Leer los datos iniciales
data = read_csv(DATA_FILE)
rows = to_DataTable(data)

def alert(error: str) → ft.AlertDialog:
    """Mostrar un diálogo de alerta con un mensaje de error."""
    return ft.AlertDialog(title=ft.Text(error))

def main(page: ft.Page):

```

```

# Configuración inicial de la página
page.title = "Tarea 4 - Jorge Juarez"
page.theme_mode = ft.ThemeMode.LIGHT
page.scroll = ft.ScrollMode.AUTO
page.vertical_alignment = ft.MainAxisAlignment.CENTER
page.horizontal_alignment = ft.CrossAxisAlignment.CENTER

# AppBar y título
title_appbar = ft.Text("Productos disponibles",
theme_style=ft.TextThemeStyle.HEADLINE_SMALL)
appbar = ft.AppBar(
    title=title_appbar,
    center_title=True,
    leading=ft.Image(src="images/wild-fork-logo.png"),
    leading_width=180
)

# Componentes de la interfaz de usuario
id_register = ft.Text("", theme_style=ft.TextThemeStyle.BODY_LARGE)
id_search = ft.TextField(label="ID del producto", icon=ft.icons.GRID_3X3)
data_register = ft.Text("", weight=ft.FontWeight.BOLD)
# Función para buscar un registro
def go_register(e):
    try:
        product_id = int(id_search.value)
        reg_info = search_register(data, product_id)
        if reg_info:
            mostrar_detalles_producto(reg_info)
        else:
            page.open(alert("No se encuentra un producto con el ID
ingresado o el producto está agotado"))
    except ValueError:
        page.open(alert("Valor ingresado incorrectamente"))
    except Exception as ex:
        print(f"Error inesperado: {ex}")

def mostrar_detalles_producto(reg_info):
    """Muestra los detalles del producto seleccionado."""
    title_appbar.value = "Compra de producto"
    id_register.value = f"Detalles del producto: {id_search.value}"
    data_register.value = str(reg_info)
    page.bottom_appbar = register_bottombar
    page.scroll = None
    page.controls.pop()
    page.add(register)
    page.update()

# Barra inferior en la página de inicio
home_bottombar = ft.BottomAppBar(

```

```

        content=ft.Row(
            controls=[
                id_search,
                ft.FilledButton(text="Detalles",
icon=ft.icons.SAVED_SEARCH_ROUNDED, on_click=go_register)
            ],
            vertical_alignment=ft.CrossAxisAlignment.CENTER,
            alignment=ft.MainAxisAlignment.SPACE_EVENLY
        )
    )

# Función para regresar a la página principal
def go_home(e):
    actualizar_datos()
    title_appbar.value = "Productos disponibles"
    page.scroll = ft.ScrollMode.AUTO
    page.bottom_appbar = home_bottombar
    page.controls.pop()
    page.add(data_table)
    page.update()

# Función para actualizar los datos de la tabla
def actualizar_datos():
    global data, rows
    data = read_csv(DATA_FILE)
    rows = to_DataTable(data)
    data_table.rows = rows

# Función para procesar la compra de un producto
def shop(e):
    try:
        product_id = int(id_search.value) - 1
        if int(data[product_id].stock) > 0:
            data[product_id].stock = int(data[product_id].stock) - 1
            write_csv(DATA_FILE, data)
            page.open(ft.AlertDialog(title=ft.Text(f"Producto
'{data[product_id].name}' comprado con éxito"), on_dismiss=go_home))
        else:
            page.open(alert("No hay suficiente stock del producto."))
    except Exception as ex:
        page.open(alert(f"Error en la compra: {ex}"))

# Barra inferior en la página de detalles
register_bottombar = ft.BottomAppBar(
    content=ft.Row(
        controls=[
            ft.FilledButton(text="Comprar",
icon=ft.icons.SHOPPING_BAG_OUTLINED, on_click=shop),
            ft.FilledButton(text="Regresar", icon=ft.icons.ARROW_BACK,

```

```
on_click=go_home)
    ],
    vertical_alignment=ft.CrossAxisAlignment.CENTER,
    alignment=ft.MainAxisAlignment.SPACE_EVENLY
)
)

# Tabla de productos
data_table = ft.DataTable(
    columns=[
        ft DataColumn(ft.Text("ID"), numeric=True),
        ft DataColumn(ft.Text("Nombre")),
        ft DataColumn(ft.Text("Categoria")),
        ft DataColumn(ft.Text("Peso (kg)"), numeric=True),
        ft DataColumn(ft.Text("Precio (MXN)"), numeric=True),
        ft DataColumn(ft.Text("Caducidad")),
        ft DataColumn(ft.Text("Stock"), numeric=True),
    ],
    rows=rows
)

# Contenedor de detalles del registro
register = ft.Container(
    content=ft.Column(
        controls=[
            id_register,
            ft.Container(
                content=data_register,
                padding=ft.padding.all(5),
                border=ft.border.all(2.5, "0"),
                border_radius=ft.border_radius.all(15)
            )
        ]
    )
)

# Configuración inicial de la página
page.appbar = appBar
page.bottom_appbar = home_bottombar
page.add(data_table)

ft.app(
    main,
    assets_dir="assets"
)
```

Este código permite crear la siguiente interfaz con las siguientes vistas:

Tarea 4 - Jorge Juarez

# Wild Fork

## Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	26
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5
9	Salsa de tomate	Salsas	0.5	\$30	2024-12-31	12

# ID del producto ID del producto Detalles



## Memoria compartida

En el contexto de este proyecto con Flet, el uso de **memoria compartida** permite que múltiples procesos accedan y modifiquen de manera eficiente una estructura de datos común (en este caso, una lista de productos) sin tener que duplicar los datos en la memoria de cada proceso. Esto es especialmente útil cuando trabajamos con aplicaciones multiproceso que requieren sincronización y acceso concurrente a los datos, como en la interfaz de usuario de una tienda que muestra productos y permite realizar compras en tiempo real.

### Fucionamiento técnico:

#### 1. Inicialización de la memoria compartida:

Se inicializa la memoria compartida creando una instancia de `SharedMemory`. Los datos de productos (una lista de diccionarios) son serializados con `pickle` y escritos en la memoria compartida.

#### 2. Acceso concurrente:

Cuando un proceso necesita acceder o modificar los productos, lo hace de forma segura utilizando un **semaforo** (`Semaphore`). Esto asegura que solo un proceso a la vez pueda realizar cambios en los datos, evitando conflictos.

### 3. Lectura y escritura:

Cada vez que un proceso accede a la memoria compartida, los datos son deserializados desde bytes utilizando `pickle.loads()`. Si se realizan modificaciones en los productos, los datos son actualizados en la memoria compartida con `pickle.dumps()`.

### 4. Liberación de recursos:

Al finalizar la ejecución del programa, se libera la memoria compartida con `shm.close()` y `shm.unlink()`, evitando fugas de memoria y asegurando que los recursos sean gestionados correctamente.

Entonces el script actualizado para que ocupe memoria compartida con dos procesos del sistema es el siguiente:

main\_v2.py

```
import pickle
from multiprocessing import Process, Semaphore
from multiprocessing.shared_memory import SharedMemory
from typing import Any, Union, Dict
from core import Register, read_csv, write_csv, to_DataTable
import flet as ft

# Ruta del archivo CSV
DATA_FILE = "./data/wildfork.csv"

# Crear memoria compartida para los productos
def create_shared_product_list() → SharedMemory:
    data = read_csv(DATA_FILE)
    product_list = [
        {
            'id_reg': reg.id_reg,
            'name': reg.name,
            'category': reg.category,
            'weight': reg.weight,
            'price': reg.price,
            'expiry': reg.expiry,
            'stock': reg.stock,
            'is_locked': False # Bandera para indicar si está
bloqueado
        }
        for reg in data
    ]
    serialized_data = pickle.dumps(product_list) # Serializamos la lista de
diccionarios
    shm = SharedMemory(create=True, size=len(serialized_data)) # Creamos
memoria compartida
    shm.buf[:len(serialized_data)] = serialized_data # Escribimos los datos
en la memoria compartida
    return shm
```

```

# Función para leer los datos de la memoria compartida
def read_shared_product_list(shm: SharedMemory) → Any:
    serialized_data = bytes(shm.buf[:]) # Leemos los datos serializados
    return pickle.loads(serialized_data) # Deserializamos para obtener la
lista de productos

# Función para escribir los datos en la memoria compartida
def write_shared_product_list(shm: SharedMemory, products: Any) → None:
    serialized_data = pickle.dumps(products)
    shm.buf[:len(serialized_data)] = serialized_data # Actualizamos la
memoria compartida con los nuevos datos

# Función para acceder a los detalles del producto con bloqueo
def access_product(shm: SharedMemory, sem, product_id: int) → Union[bool,
Dict]:
    with sem:
        products = read_shared_product_list(shm)
        product = products[product_id]
        if product['is_locked']:
            return False # El producto está bloqueado por otra
instancia
        else:
            product['is_locked'] = True
            products[product_id] = product
            write_shared_product_list(shm, products) # Actualizamos
la memoria compartida
            return product # Devuelve el producto para mostrar los
detalles

# Función para desbloquear un producto después de la compra o regreso
def unlock_product(shm: SharedMemory, sem, product_id: int) → None:
    with sem:
        products = read_shared_product_list(shm)
        product = products[product_id]
        product['is_locked'] = False
        products[product_id] = product
        write_shared_product_list(shm, products) # Actualizamos la
memoria compartida

def alert(msj: str) → ft.AlertDialog:
    return ft.AlertDialog(title=ft.Text(msj))

def main(page: ft.Page, shm: SharedMemory, sem):
    # Configuración inicial de la página
    page.title = "Tarea 4 - Jorge Juarez"
    page.theme_mode = ft.ThemeMode.LIGHT
    page.scroll = ft.ScrollMode.AUTO
    page.vertical_alignment = ft.MainAxisAlignment.CENTER

```

```

page.horizontal_alignment = ft.CrossAxisAlignment.CENTER

# AppBar y título
title_appbar = ft.Text("Productos disponibles",
theme_style=ft.TextThemeStyle.HEADLINE_SMALL)

appbar = ft.AppBar(
    title=title_appbar,
    center_title=True,
    leading=ft.Image(src="images/wild-fork-logo.png"),
    leading_width=180
)

id_register = ft.Text("", theme_style=ft.TextThemeStyle.BODY_LARGE)
id_search = ft.TextField(label="ID del producto",
icon=ft.icons.GRID_3X3)
data_register = ft.Text("", weight=ft.FontWeight.BOLD)

def go_register(e):
    try:
        product_id = int(id_search.value) - 1
        product = access_product(shm, sem, product_id)
        if product:
            mostrar_detalles_producto(product, product_id)
        else:
            page.open(alert("El producto está siendo
accedido por otra instancia"))
    except ValueError:
        page.open(alert("Valor ingresado incorrectamente"))
    except IndexError:
        page.open(alert(f"Producto con el ID: {id_search.value}
no encontrado"))
    except Exception as ex:
        page.open(alert(f"Error inesperado: {ex}"))

def mostrar_detalles_producto(product, product_id):
    """Muestra los detalles del producto seleccionado."""
    title_appbar.value = "Compra de producto"
    id_register.value = f"Detalles del producto: {product['id_reg']}"
    data_register.value = """
Nombre: {product['name']}
Categoría: {product['category']}
Peso: {product['weight']} kg
Precio: {product['price']} MXN
Caducidad: {product['expiry']}
Stock: {product['stock']}
"""

    page.bottom_appbar = register_bottombar(product_id)
    page.scroll = None

```

```

page.controls.pop()
page.add(register)
page.update()

# Barra inferior en la página de inicio
home_bottombar = ft.BottomAppBar(
    content=ft.Row(
        controls=[
            id_search,
            ft.FilledButton(text="Detalles",
icon=ft.icons.SAVED_SEARCH_ROUNDED, on_click=go_register)
        ],
        vertical_alignment=ft.CrossAxisAlignment.CENTER,
        alignment=ft.MainAxisAlignment.SPACE_EVENLY
    )
)

def go_home(product_id=None):
    if product_id is not None:
        unlock_product(shm, sem, product_id)
    update_data()
    title_appbar.value = "Productos disponibles"
    page.scroll = ft.ScrollMode.AUTO
    page.bottom_appbar = home_bottombar
    page.controls.pop()
    page.add(data_table)
    page.update()

def update_data():
    """Actualiza la tabla de productos con los datos en memoria
compartida."""
    products = read_shared_product_list(shm)
    data_table.rows = to_DataTable([Register(**p) for p in products])

def shop(product_id):
    with sem:
        products = read_shared_product_list(shm)
        if int(products[product_id]['stock']) > 0:
            product_copy = products[product_id].copy()
            product_copy['stock'] = int(product_copy['stock']) - 1
            products[product_id] = product_copy
            write_shared_product_list(shm, products)
            write_csv(DATA_FILE, [Register(**p) for p in products])
            page.open(ft.AlertDialog(title=ft.Text(f"Producto
'{products[product_id]['name']}' comprado con éxito"), on_dismiss=lambda e:
go_home(product_id))))
        else:
            page.open(alert("No hay suficiente stock del producto."))

```

```

def register_bottombar(product_id):
    return ft.BottomAppBar(
        content=ft.Row(
            controls=[
                ft.FilledButton(text="Comprar",
icon=ft.icons.SHOPPING_BAG_OUTLINED, on_click=lambda e: shop(product_id)),
                ft.FilledButton(text="Regresar", icon=ft.icons.ARROW_BACK,
on_click=lambda e: go_home(product_id))
            ],
            vertical_alignment=ft.CrossAxisAlignment.CENTER,
            alignment=ft.MainAxisAlignment.SPACE_EVENLY
        )
    )

data_table = ft.DataTable(
    columns=[
        ft DataColumn(ft.Text("ID"), numeric=True),
        ft DataColumn(ft.Text("Nombre")),
        ft DataColumn(ft.Text("Categoría")),
        ft DataColumn(ft.Text("Peso (kg)"), numeric=True),
        ft DataColumn(ft.Text("Precio (MXN)"), numeric=True),
        ft DataColumn(ft.Text("Caducidad")),
        ft DataColumn(ft.Text("Stock"), numeric=True)
    ]
)

register = ft.Container(
    content=ft.Column(
        controls=[
            id_register,
            ft.Container(
                content=data_register,
                padding=ft.padding.all(5),
                border=ft.border.all(2.5, "0"),
                border_radius=ft.border_radius.all(15)
            )
        ]
    )
)

update_data()
page.appbar = appbar
page.bottom_appbar = home_bottombar
page.add(data_table)

def run_flet_app(shm: SharedMemory, sem) → None:
    ft.app(target=lambda page: main(page, shm, sem), assets_dir="assets")

```

```
if __name__ == "__main__":
    shm = create_shared_product_list()
    sem = Semaphore(1)

    p1 = Process(target=run_flet_app, args=(shm, sem))
    p2 = Process(target=run_flet_app, args=(shm, sem))

    p1.start()
    p2.start()

    p1.join()
    p2.join()

    shm.close()
    shm.unlink()
```

## Funcionamiento (pantallazos)

FedoraOS [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

12 de sep 13:19

Tarea 4 - Jorge Juarez

(Tarea4) jorgeirzz@fedora:~/Documentos/Tarea4 — /usr/bin/py...

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

#  ID del producto  Detalles

FedoraOS [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

12 de sep 13:20

Tarea 4 - Jorge Juarez

(Tarea4) jorgeirzz@fedora:~/Documentos/Tarea4 — /usr/bin/py...

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

Valor ingresado incorrectamente  ID del producto  Detalles

#  error

FedoraOS [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

12 de sep 13:21

Tarea 4 - Jorge Juarez

```
jorgejrz@fedora:~/Documentos/Tarea4 — /usr/bin/py... python main.py
package:media_kit.libs_linux registered.
package:media_kit.libs_linux registered.
```

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

Producto con el ID: 10000000 no encontrado

ID del producto: 10000000

Detalles

Tarea 4 - Jorge Juarez

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

ID del producto: 10000000

Detalles

CTRL DERECHA

FedoraOS [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

12 de sep 13:22

```
jorgejrz@fedora:~/Documentos/Tarea4 — /usr/bin/py... python main.py
package:media_kit.libs_linux registered.
package:media_kit.libs_linux registered.
```

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

ID del producto: 10000000

Detalles

Tarea 4 - Jorge Juarez

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

ID del producto: 10000000

Detalles

CTRL DERECHA

FedoraOS [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

12 de sep 13:23

Tarea4 Jorge Juarez

(Tarea4) jorgejrzz@fedora:~/Documentos/Tarea4 — /usr/bin/py... x

jorgejrzz@fedora:~/Documentos/Tarea4\$ python main.py

package:media\_kit.libs\_linux registered.

package:media\_kit.libs\_linux registered.

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

Detalles del producto: 8

Nombre: Camarones jumbo  
Categoría: Mariscos  
Peso: 0.5 kg  
Precio: 180 MXN  
Caducidad: 2024-11-15  
Stock: 5

Comprar Regresar

ID del producto: 1000000

Detalles

FedoraOS [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

12 de sep 13:23

Tarea4 Jorge Juarez

(Tarea4) jorgejrzz@fedora:~/Documentos/Tarea4 — /usr/bin/py... x

jorgejrzz@fedora:~/Documentos/Tarea4\$ python main.py

package:media\_kit.libs\_linux registered.

package:media\_kit.libs\_linux registered.

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

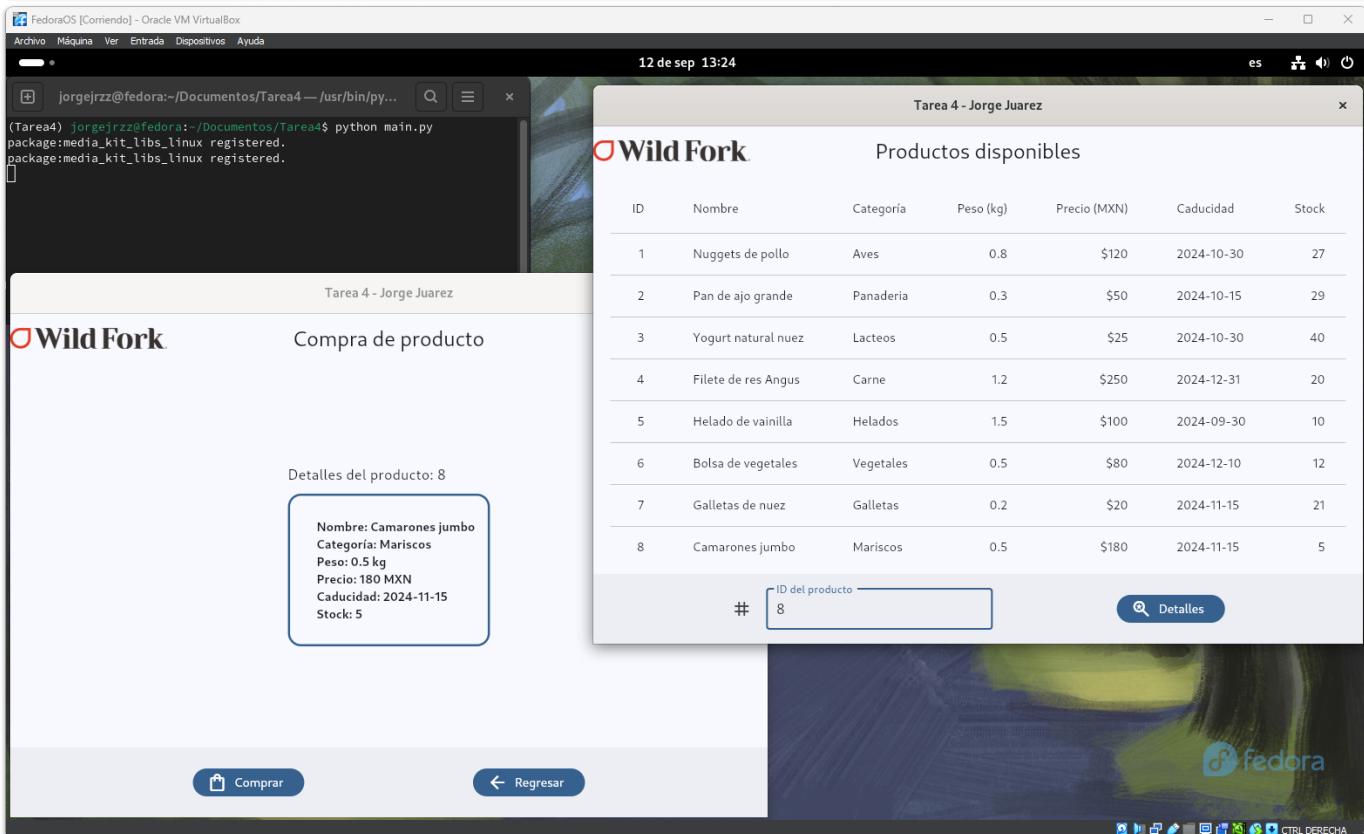
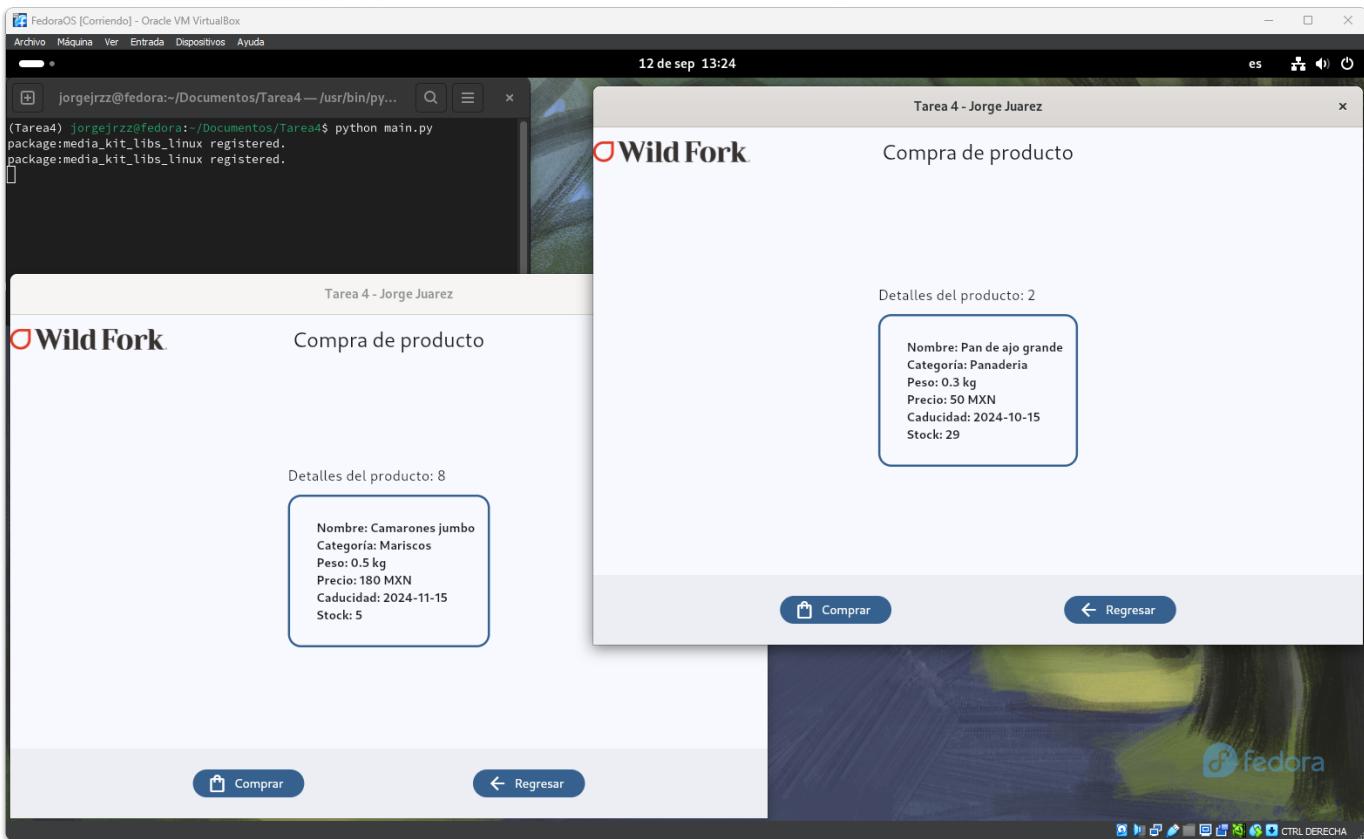
Detalles del producto: 8

Nombre: Camarones jumbo  
Categoría: Mariscos  
Peso: 0.5 kg  
Precio: 180 MXN  
Caducidad: 2024-11-15  
Stock: 5

Comprar Regresar

ID del producto: 2

Detalles



FedoraOS [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

12 de sep 13:24

jorgejrzz@fedora:~/Documentos/Tarea4 — /usr/bin/py... x

(Tarea4) jorgejrzz@fedora:~/Documentos/Tarea4\$ python main.py

package:media\_kit.libs\_linux registered.

package:media\_kit.libs\_linux registered.

Tarea 4 - Jorge Juarez

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filetes de pescado					20
5	Helado vainilla					10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

El producto está siendo accedido por otra instancia

# ID del producto

8

Detalles

Comprar

Regresar

Wild Fork

Compra de producto

Detalles del producto: 8

Nombre: Camarones jumbo  
Categoría: Mariscos  
Peso: 0.5 kg  
Precio: 180 MXN  
Caducidad: 2024-11-15  
Stock: 5

Comprando...

Tarea 4 - Jorge Juarez

CTRL DERECHA

FedoraOS [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

12 de sep 13:25

jorgejrzz@fedora:~/Documentos/Tarea4 — /usr/bin/py... x

(Tarea4) jorgejrzz@fedora:~/Documentos/Tarea4\$ python main.py

package:media\_kit.libs\_linux registered.

package:media\_kit.libs\_linux registered.

Tarea 4 - Jorge Juarez

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filetes de pescado					20
5	Helado vainilla					10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

El producto está siendo accedido por otra instancia

# ID del producto

8

Detalles

Comprando...

Producto 'Camarones jumbo' comprado con éxito

Stock: 5

Comprando...

Regresar

Wild Fork

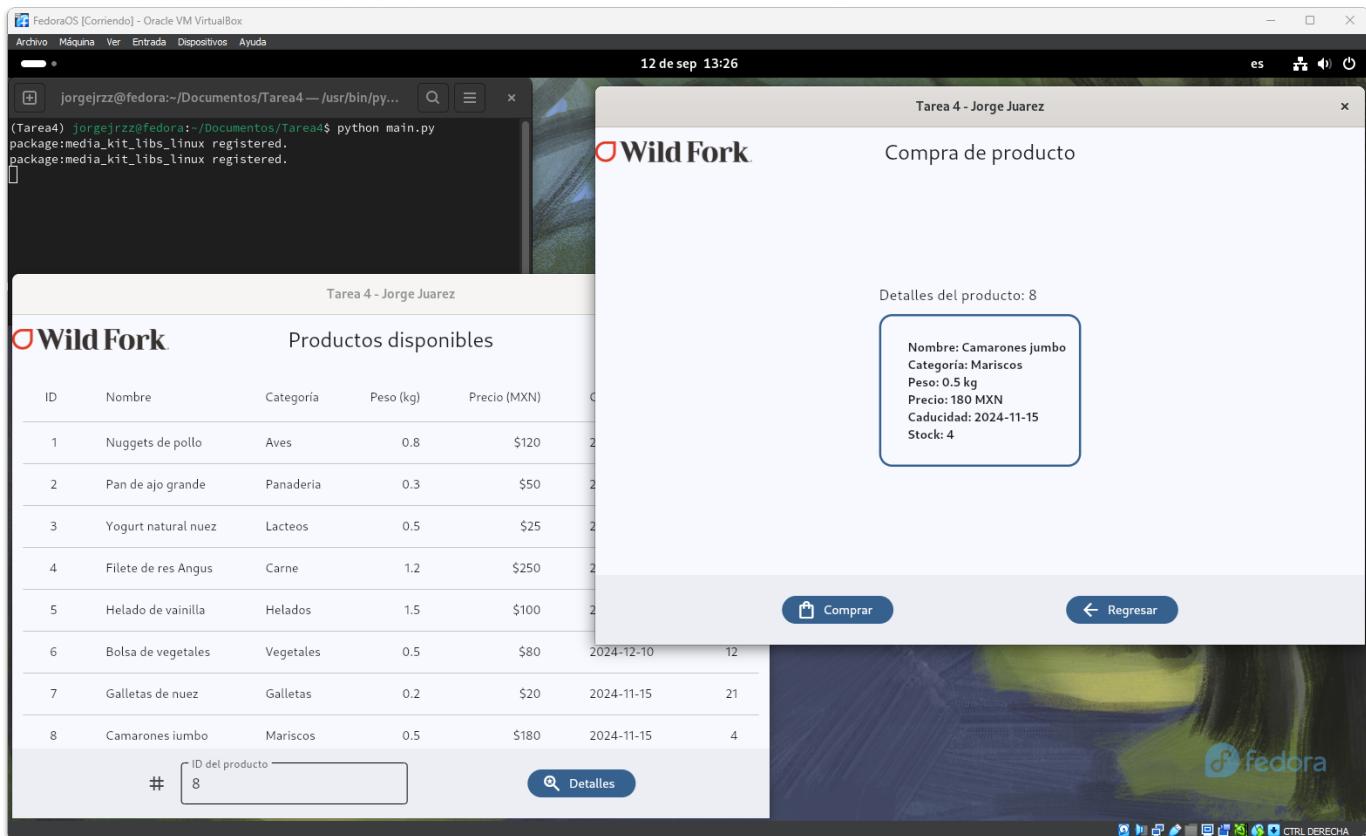
Compra de producto

Detalles del producto: 8

Comprando...

Tarea 4 - Jorge Juarez

CTRL DERECHA



## Comprobación del uso de memoria compartida

En Python, la biblioteca `multiprocessing` ofrece la capacidad de compartir datos entre procesos mediante el uso de **memoria compartida**. Esto es especialmente útil para proyectos multiproceso, como este, donde varias instancias de una aplicación deben acceder a los mismos datos sin tener que duplicarlos en la memoria de cada proceso. En esta tarea, se utilizó la memoria compartida para almacenar y acceder de manera eficiente a una lista de productos, permitiendo que múltiples procesos modifiquen estos datos de forma sincronizada.

## ¿Cómo funciona la memoria compartida en `multiprocessing`?

`multiprocessing.shared_memory` es una herramienta que permite crear una región de memoria accesible para varios procesos. En este proyecto, hemos utilizado esta característica para almacenar una lista de diccionarios que representan los productos del negocio WildFork. La estructura de datos compleja (lista de diccionarios) no se puede almacenar directamente en la memoria compartida, por lo que se utiliza la serialización mediante la biblioteca `pickle`. Esto convierte los datos en una secuencia de bytes que puede ser almacenada en memoria compartida y luego deserializada cuando se accede a ella.

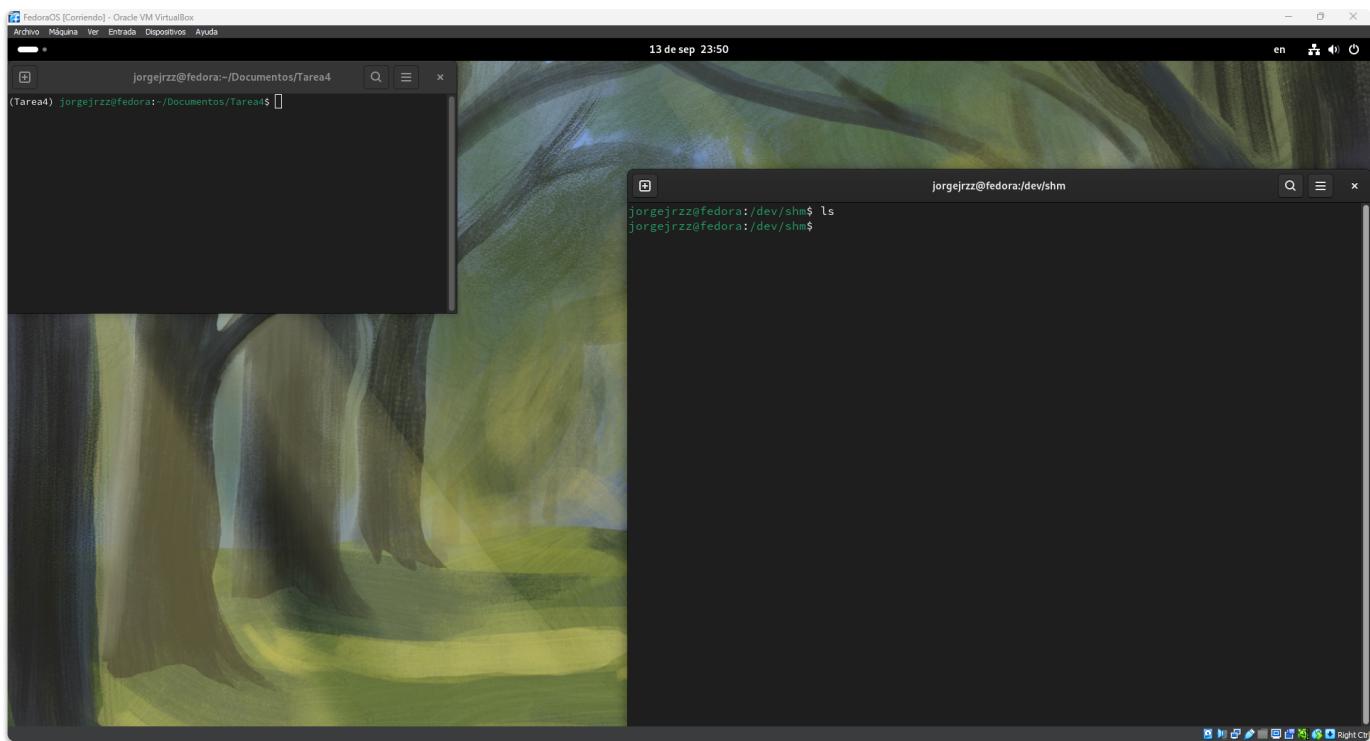
## El comando `ipcs -m`

El comando `ipcs` en sistemas UNIX y Linux se utiliza para mostrar los segmentos de memoria compartida del sistema operativo que han sido creados mediante la interfaz **System V IPC** o **POSIX**. Sin embargo, la memoria compartida creada con `multiprocessing.shared_memory` en Python no utiliza este tipo de IPC de bajo nivel. En su lugar, se basa en un mecanismo de memoria compartida que es gestionado directamente por Python y no aparece en los listados de `ipcs`. Esto significa que, aunque Python esté utilizando memoria compartida, no es visible a través de las herramientas tradicionales de monitoreo del sistema como `ipcs`. En su lugar, podemos encontrar evidencia de los segmentos de memoria compartida en la carpeta del sistema `/dev/shm`, donde se almacena la memoria compartida temporalmente.

### Ruta `/dev/shm`

En sistemas basados en Linux, la memoria compartida creada por Python a través de `multiprocessing.shared_memory` se puede observar en la ruta `/dev/shm`. Esta carpeta es utilizada por el sistema operativo para almacenar archivos temporales en memoria RAM. Por ejemplo, cuando ejecutamos el programa, podemos ver un archivo con un nombre generado automáticamente que corresponde a la región de memoria compartida utilizada por el programa. Al examinar este archivo con herramientas como `hexdump` o `xxd`, podemos visualizar el contenido binario que está almacenado en la memoria compartida.

En nuestro caso, al correr el programa, se genera un archivo en `/dev/shm` con un nombre como `psm_xxxxxx`, que contiene la información serializada de los productos. Este archivo permanece en el sistema hasta que se cierra y se desvincula la memoria compartida con `shm.unlink()`.



A screenshot of a Fedora desktop environment showing multiple windows. On the left, a terminal window shows the command 'python main\_v2.py' being run. In the center, a web browser displays a table of products from 'Wild Fork'. On the right, another terminal window shows the command 'ls' and its output. The desktop background is a painting of a forest scene.

**Terminal 1:**

```
jorgejrzz@fedora:~/Documentos/Tarea4 — /usr/bin/python...  
jorgejrzz@fedora:~/Documentos/Tarea4$ python main_v2.py  
package:media_kit_libs_linux registered.  
package:media_kit_libs_linux registered.
```

**Terminal 2:**

```
jorgejrzz@fedora:~/Documentos/Tarea4$ ls  
jorgejrzz@fedora:~/Documentos/Tarea4$ ls  
psm_073b6105  
jorgejrzz@fedora:~/Documentos/Tarea4$
```

**Web Browser:**

**Wild Fork** Products disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	25
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	19
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	B					
7						
8						
9						
10						

```

FedoraOS [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
13 de sep 23:47
[Tarea4] jorgejrzz@fedora:~/Documentos/Tarea4-- /usr/bin/python...
[Tarea4] jorgejrzz@fedora:~/Documentos/Tarea4$ python main_v2.py
package:media_kit_libs_linux registered.
package:media_kit_libs_linux registered.

Wild Fork
Productos disponibles

ID Nombre Categoría Peso(kg) Precio(MXN) Caducidad Stock
1 Nuggets de pollo Aves 0.8 $20 2024-10-30 25
2 Pan de ajo grande Panadería
3 Yogurt natural nuez Lacteos
4 Filete de res Angus Carne
5 Helado de vainilla Helados
6 Bolsa de vegetales Vegetales
7 Galletas de nuez Galletas
8 Camarones jumbo Mariscos
9 Salsa de tomate Salsas
10 Pan de ajo chico Panadería 0.1 $20 2024-10-15 29

# ID del producto

# ID del producto

jorgejrzz@fedora:~/dev/shm$ ls
jorgejrzz@fedora:~/dev/shm$ ls
psm_073b6185
jorgejrzz@fedora:~/dev/shm$ xxd psm_073b6185
00000000: 8094 9589 0300 0000 0000 000d 9428 7d94 .....].
00000010: 288c 0669 64f5 7265 6794 8c01 3194 8c04 (..._id...).
00000020: 6e61 6d65 948c 7567 6765 7473 2064 name...Nuggets d
00000030: 6520 706f 6cc6 6f94 8c08 6361 7465 676f e polle...cate...
00000040: 7279 948c 0441 7665 7394 8c09 7765 6967 ry...Aves...wei...
00000050: 6874 948c 0330 2e33 948c 0570 7269 6365 ht...0...price...
00000060: 948c 0331 3230 948c 0665 7870 6972 7994 .....120...expiry...
00000070: 8c0a 3230 3234 2d31 302d 3330 948c 0573 .....2024-10-30...s...
00000080: 746f 6361 948c 0232 3594 8c09 6973 5f6e tock...25...is_l...
00000090: 6f63 6b65 6494 8975 7d94 2868 028c 0132 ocked...u...h...2
000000a0: 9468 048c 1150 616 2064 6520 616a 6f20 h...Pan de ajo...
000000b0: 6772 616c 6465 9468 088c 0330 2e33 9468 0a8c grande...Panad...
000000c0: 6572 6961 9468 088c 0330 2e33 9468 0a8c eria.h...0.3...
000000d0: 0235 3094 680c 8c0a 3230 3234 2d31 302d 50...2024-10...
000000e0: 3135 9468 0e8c 0232 3994 6810 8975 7d94 15...h...29...h...u...
000000f0: 2868 028c 0133 9468 048c 1359 6f67 7572 (h...3...h...Yogur...
00000100: 7420 616c 7475 7261 6c20 6e75 6575 9468 t natural...nuez.h...
00000110: 068c 074c 6163 7465 6f73 9468 088c 0330 ...Lacteos.h...0
00000120: 2e35 9468 0a8c 0232 3594 680c 8c0a 3230 .5...h...25...h...20
00000130: 6874 948c 0330 2d31 302d 3330 9468 0e8c 0234 3094 24-10-30...h...40...
00000140: 6810 8975 7d94 2868 028c 0134 9468 048c h...u...h...4...
00000150: 1346 9696 6574 6465 2072 6573 2041 .Filete de res A...
00000160: 6e67 5753 9468 0680 0543 6172 6e65 9468 ngus.h...Carne.h...
00000170: 088c 0331 2e32 9468 0a8c 0332 3530 9468 .....1.2...h...250.h...
00000180: 0c8c 0a32 3032 342d 3132 2d33 3194 680e ...2024-12-31...h...
00000190: 8c02 3133 9468 1089 757d 9428 6802 8c01 .....19...h...u...h...
000001a0: 3594 6804 8c12 4865 6c61 646f 2064 6520 5h...Helado de...
000001b0: 7661 6961 696c 6c61 9468 068c 0231 3530 9468 vainilla.h...Hel...
000001c0: 6164 6f73 9468 088c 0231 2e35 9468 0a8c ados.h...1.5...
00000200: 6120 6465 2076 6567 6574 616c 6573 9468 a de...vegetales.h...
00000210: 065c 0956 6567 6574 616c 6573 9468 088c ...Vegetales.h...
00000220: 0330 2e35 9468 0a8c 0230 3094 680c 8c0a .....0.5...h...80...h...
00000230: 3230 3234 2d31 322d 3130 9468 088c 0231 2024-12-10...h...1
00000240: 3294 6810 8975 7d94 2868 028c 0137 9468 2.h...u...h...7...
00000250: 048c 1947 616c 6c65 7461 7320 6465 2064 ...Galletas de n...
00000260: 7565 7a94 6896 8c09 4761 6c61 6574 6173 uez...Galletas...
00000270: 0468 088c 0330 2e32 9468 0a8c 0232 3094 .....h...0.2...h...20...
00000280: 680c 8c0a 3230 3234 2d31 312d 3135 9468 .....h...2024-11-15...h...
00000290: 0e8c 0232 3194 6610 8975 7d94 2868 028c .....21...h...u...h...
000002a0: 0138 9468 048c 0f43 616d 6172 6f6e 6573 .....h...Camarones...
000002b0: 206a 756d 626f 946c 068c 084d 6172 6973 jumbo...Maris...
000002c0: 636f 7394 6808 8c03 302e 3594 680a 8c03 cos...h...0.5...
000002d0: 1138 3094 680c 3230 3234 2d31 312d 180...2024-11-...
000002e0: 1135 9468 0e88 2261 1089 757d 9428 6802 15...h...h...u...h...
000002f0: 4c01 3994 6804 8c0f 5361 6c73 6120 6465 .....9...
00000300: 2074 6f6d 6174 6594 6806 8c06 5361 6c73 ...Salsa de...
00000310: 6173 9468 088c 0330 2e35 9468 0a8c 0233 as.h...0.5...h...3
00000320: 3094 680c 8c0a 3230 3234 2d31 322d 3331 0.h...2024-12-31
00000330: 9468 0e8c 0231 3294 6810 8975 7d94 2868 .....h...12...h...u...h...
00000340: 028c 0231 3094 6804 8c10 5061 6e20 6465 .....10...Pan de...
00000350: 2091 6a6f 2063 6869 636f 9468 068c 0950 ajo chico...h...P
00000360: 616e 6164 6572 6961 9468 088c 0330 2e31 anaderia.h...0.1
00000370: 9468 0a8c 0232 3094 680c 8c0a 3230 3234 .....h...20...h...2024
00000380: 2d31 302d 3135 9468 0e8c 0232 3994 6810 .....-10...h...29...h...
00000390: 8975 652e .....ue.

jorgejrzz@fedora:~/dev/shm$ ls
jorgejrzz@fedora:~/dev/shm$
```

## shared\_memory vs Manager

La diferencia principal entre usar `shared_memory` y `Manager` (uso en la versión uno) en Python radica en cómo cada uno gestiona la memoria compartida y la eficiencia en el acceso a los datos por múltiples procesos.

### 1. Shared Memory (`multiprocessing.shared_memory`):

- **Eficiencia:** `shared_memory` permite compartir directamente bloques de memoria entre procesos sin intermediarios, lo que lo hace más eficiente en términos de velocidad y consumo de recursos. No es necesario duplicar los datos en la memoria de cada proceso, ya que todos acceden a la misma región de memoria.
- **Control manual:** El uso de `shared_memory` requiere una gestión más manual, ya que el programador debe controlar la serialización/deserialización de datos complejos (por ejemplo, usando `pickle`), además de encargarse del cierre y la eliminación de la memoria compartida una vez que ya no sea necesaria.
- **Adecuado para datos simples:** Si bien es altamente eficiente para compartir arrays o estructuras de datos simples (por ejemplo, con NumPy), cuando se trata de datos complejos como listas de diccionarios, es necesario realizar serialización, lo que agrega cierta complejidad.
- **No visible en `ipcs`:** La memoria compartida creada mediante este método no se gestiona mediante la capa de IPC (System V o POSIX), por lo que no se puede monitorear con herramientas como `ipcs`. Sin embargo, es visible en sistemas Linux en la carpeta `/dev/shm`.

## 2. Manager (`multiprocessing.Manager`):

- **Simplicidad:** `Manager` es más sencillo de usar, ya que abstrae muchos de los detalles de la concurrencia y el acceso a los datos. Permite compartir fácilmente estructuras de datos complejas (como listas, diccionarios, etc.) entre procesos, sin necesidad de preocuparse por la serialización o el manejo explícito de la memoria.
- **Menor eficiencia:** Aunque `Manager` es más fácil de usar, es menos eficiente que `shared_memory`. Esto se debe a que los datos no se comparten directamente en memoria; en su lugar, se gestiona a través de un servidor proxy que coordina el acceso a los datos. Esto introduce cierta sobrecarga en el sistema, especialmente cuando hay muchas operaciones concurrentes.
- **Adecuado para datos complejos:** `Manager` es ideal cuando se necesita compartir estructuras de datos complejas y no se busca la máxima eficiencia en términos de memoria o velocidad.
- **Visibilidad y monitoreo:** Al usar `Manager`, los datos no están directamente en memoria compartida del sistema operativo, por lo que tampoco aparecerán en herramientas como `ipcs`.

## Conclusión

En esta actividad se implementó una aplicación de gestión de productos utilizando `Flet` para la interfaz gráfica y Python para la lógica subyacente, con un enfoque particular en el manejo de concurrencia mediante **memoria compartida**. A diferencia del enfoque anterior que utilizaba `Manager` y `Lock`, en esta versión se empleó `multiprocessing.shared_memory` junto con semáforos para gestionar el acceso concurrente a una lista de productos. Esto permitió que

múltiples instancias del programa accedan a los datos de manera eficiente, sin duplicación en la memoria, y sincronicen sus acciones, como la compra y visualización de productos, garantizando la integridad de los datos en tiempo real.

El uso de la memoria compartida y la serialización de datos complejos mediante `pickle` fue clave para permitir el acceso a estructuras de datos no triviales, como listas de diccionarios, en la memoria compartida. La implementación de semáforos permitió la sincronización de procesos, evitando condiciones de carrera y asegurando que solo un proceso a la vez pudiera modificar un producto. Además, se garantizó la actualización dinámica del inventario, sincronizando los cambios entre los procesos y el archivo CSV, que actúa como una base de datos persistente.

La experiencia de implementar memoria compartida y procesos concurrentes resulta esencial para desarrollar sistemas eficientes y escalables. Este conocimiento es fundamental para aplicaciones que requieren una alta concurrencia y acceso sincronizado a datos compartidos, como sistemas de inventarios en tiempo real, servidores web y entornos de alta demanda de recursos.

Finalmente, Python es un lenguaje muy poderoso, versátil, y también una de las características clave que tiene, es que es muy fácil de *leer*, esto claro, a costa de eficiencia computacional comparado con otros lenguajes de alto nivel, sin embargo, se permite desarrollar programas con distintas características de distintas formas y con distintas bibliotecas y funcionalidades nativas del lenguaje, entonces resulta interesante preguntarse que vamos a sacrificar, más recursos, pero un código más limpio, o más eficiencia pero un código más complejo de revisar en un futuro.