Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Ingeniería en computación

Seminario de solución de problemas de sistemas operativos

NRC 164138

Sección D07

Profesor Javier Rosales Martínez

Diego Armando Sánchez Rubio 217570609

Reporte Práctica 8.

Antecedentes.

Esta es la ultima de la prácticas de memoria, en esta ocasión la modificación que recibirá es la de tener los 4 algoritmos al mismo tiempo en el programa, así cómo que tenga la opción de agregar bloque o segmentos de memoria e indicar si están disponibles u ocupados, su posición, al igual con los archivos preguntando su tamaño y posición, el programa tiene que resolver con cada uno de los algoritmos.

Metodología.

En este caso no me enfocaré en cómo se realizó el algoritmo para cada uno de los métodos porque ya se analizó en los reportes pasados, simplemente analizaré los nuevos cambios.

La función que se revisará ahora es agregar_bloque, encargada de permitir al usuario añadir nuevos bloques de memoria desde una ventana emergente.

Primero, se crea una nueva ventana (Toplevel) donde se piden los datos necesarios: el tamaño del bloque en KB, si el bloque está disponible u ocupado (usando una casilla de verificación), y la posición donde se agregará (al inicio o al final, usando botones de opción).

Al presionar el botón Agregar, se intenta convertir el valor del tamaño a entero. Si es válido, se crea un diccionario que representa el bloque con su tamaño y disponibilidad. Dependiendo de la opción elegida, se inserta al inicio o al final de la lista de bloques (self.bloques).

Luego se cierra la ventana emergente y se actualiza gráficamente la memoria con dibujar_bloques. Si el tamaño ingresado no es válido, se muestra un mensaje de error.

```python
def agregar_bloque(self):
    top = tk.Toplevel(self.root)
    top.title("Agregar Bloque de Memoria")

    tk.Label(top, text="Tamaño (KB):").grid(row=0, column=0)
    entry_tam = tk.Entry(top)
    entry_tam.grid(row=0, column=1)

    disponible_var = tk.BooleanVar(value=True)
    tk.Checkbutton(top, text="Disponible", variable=disponible_var).grid(row=1, column=0, columnspan=2)

    posicion_var = tk.StringVar(value="final")
    ttk.Radiobutton(top, text="Al inicio", variable=posicion_var, value="inicio").grid(row=2, column=0)
    ttk.Radiobutton(top, text="Al final", variable=posicion_var, value="final").grid(row=2, column=1)

    def agregar():
        try:
            tam = int(entry_tam.get())
            bloque = {"tam": tam, "disponible": disponible_var.get()}
            if posicion_var.get() == "inicio":
                self.bloques.insert(0, bloque)
            else:
                self.bloques.append(bloque)
            top.destroy()
            self.dibujar_bloques()
        except ValueError:
            messagebox.showerror("Error", "Tamaño inválido")

    ttk.Button(top, text="Agregar", command=agregar).grid(row=3, column=0, columnspan=2)
```

La función agregar_archivo_virtual permite al usuario ingresar manualmente un archivo desde una ventana emergente, simulando que es un archivo virtual.

Dentro de la ventana, se le solicita al usuario ingresar el nombre del archivo y su tamaño en KB. Además, puede elegir la posición en la que se agregará el archivo en la lista: al inicio o al final.

Cuando el usuario da clic en el botón Agregar, se intenta convertir el tamaño ingresado a entero. Si es válido, se crea una tupla con el nombre y tamaño del archivo. Esta tupla se inserta en la lista de archivos (self.archivos) en la posición elegida.

Si los datos son correctos, se cierra la ventana emergente. Si el tamaño no es un número válido, se muestra un mensaje de error.

```
def agregar_archivo_virtual(self):
    top = tk.Toplevel(self.root)
    top.title("Agregar Archivo Virtual")

    tk.Label(top, text="Nombre:").grid(row=0, column=0)
    entry_nombre = tk.Entry(top)
    entry_nombre.grid(row=0, column=1)

    tk.Label(top, text="Tamaño (KB):").grid(row=1, column=0)
    entry_tam = tk.Entry(top)
    entry_tam.grid(row=1, column=1)

    posicion_var = tk.StringVar(value="final")
    ttk.Radiobutton(top, text="Al inicio", variable=posicion_var, value="inicio").grid(row=2, column=0)
    ttk.Radiobutton(top, text="Al final", variable=posicion_var, value="final").grid(row=2, column=1)

    def agregar():
        try:
            nombre = entry_nombre.get()
            tam = int(entry_tam.get())
            if posicion_var.get() == "inicio":
                self.archivos.insert(0, (nombre, tam))
            else:
                self.archivos.append((nombre, tam))
            top.destroy()
        except ValueError:
            messagebox.showerror("Error", "Tamaño inválido")

    ttk.Button(top, text="Agregar", command=agregar).grid(row=3, column=0, columnspan=2)
```

Conclusiones.

Esta práctica no fue tan complicada porque ya tenía los 4 algortimos listos desde antes, así que no hubo problemas por esa parte, prácticamente todo fue hacer los nuevos requerimientos de la práctica y la verdad no fue tan difícil, fue entretenido y al igual que otras fue didáctico.

Referencias.

Eslabon. Estación para Laboratorios Online. (n.d.).

https://labvirtual.webs.upv.es/Best_Fit.htm

Spasojevic, A. (2025, March 27). *¿Qué es la asignación de primer ajuste?* phoenixNAP IT Glossary. https://phoenixnap.es/glosario/asignaci%C3%B3n-de-primer-ajuste

tkinter — Python interface to Tcl/Tk. (n.d.). Python Documentation. https://docs.python.org/3/library/tkinter.html