Sistemas Operativos
Práctica 1

1. Archivos y estructuras.

Se pretende realizar un programa que gestione la información de animales en una veterinaria. Suponga que se desea almacenar los datos de perros que llegan al sitio. La información a almacenar es la siguiente:

- Nombre. Cadena de máximo 32 caracteres.
- Tipo. [perro, gato, tortuga, etc] Cadena de máximo 32 caracteres.
- Edad [años]. Entero de 32 bits.
- Raza. Cadena de máximo 16 caracteres.
- Estatura [cm]. Entero de 32 bits.
- Peso [Kg]. Real de 32 bits.
- Sexo [H/M]. 1 caracter.

Al ejecutar el programa se muestra el siguiente menú, ingresando a cada opción al ingresar el número y presionar *enter*:

1. **Ingresar registro**. Al ingresar pide uno a uno, los campos de un registro.
2. **Ver registro**. Al ingresar muestra el número de registros presentes y solicita el número del registro a ver. Valida que el número exista. A partir de este menú es posible abrir la historia clínica de la mascota, que consiste en un archivo de texto plano almacenado en disco. El archivo debe ser abierto de forma automática en un editor de texto del sistema para poder ser visto y/o editado.
3. **Borrar registro**. Al ingresar muestra el número de registros presentes y solicita el número del registro a borrar. El registro es borrado del archivo, por lo que el archivo debe reducir su tamaño.
4. **Buscar registro**. Solicita una cadena de caracteres a buscar en los campos *nombre* de los registros. Muestra todos los registros que coincidan completamente con el nombre. No se distingue mayúsculas de minúsculas.
5. **Salir**.

1.1. Consideraciones.

- Agrupar la información de un registro en la estructura *dogType*.
- Hacer uso de punteros y de memoria dinámica (malloc() - free()).
- Por cada opción ejecutada, siempre se debe dar un mensaje de confirmación y solicitar cualquier tecla para continuar, antes de volver al menú principal.
- Los datos ingresados siempre son almacenados en un archivo llamado *dataDogs.dat* que reside en la carpeta desde donde se ejecute el programa.
- Almacenar la estructura de forma binaria, no texto y deberá residir en disco, NO en memoria. La cantidad de memoria empleada por el proceso no deberá soportar los 100MB.
- Se debe implementar una tabla *hash* de al menos 1000 entradas para la búsqueda de una estructura en el archivo.
- Se deberá generar un archivo de al menos 10'000.000 de estructuras, generados de forma aleatoria mediante un programa independiente. Los nombres serán seleccionados de forma aleatoria de una lista de mínimo 1000 nombres (archivo adjunto).
- Al ingresar un nuevo registro, se deberá actualizar el archivo *dataDogs.dat*.
- Entrega: Archivo fuente con *main* en **p1-dogProgram.c** (se pueden tener más archivos) y archivo LEEME dentro de una carpeta con los nombres que aparecen en el correo para cada integrante. Pej: **capedrazab-capedrazab**. Este archivo o carpeta se entregará en clase.

1.2. Calificación.

Se tendrán en cuenta los siguientes aspectos para la evaluación:

- Funcionamiento del programa. Opción 1: 10 % Opción 2: 10 % Opción 3: 10 % Opción 4: 10 %
- Rapidez de la búsqueda (tabla hash, generación de archivo dat). 10 %
- Código limpio. 10 % (modular, tabulaciones, comentarios - básicos, declaración de constantes, etc.)
- Sustentación (se selecciona a cualquier integrante): 40 %.