UNIVERSIDAD NACIONAL DE COLOMBIA

Sistemas operativos: Práctica #2
Veterinaria Pet Family-Listado de funciones

Valentina Bueno Valles [a] Andrés Felipe Moya Rodríguez [b]
a Facultad de Ingeniería, Universidad Nacional de Colombia, Bogotá, Colombia. vbueno@unal.edu.co
b Facultad de Ingeniería, Universidad Nacional de Colombia, Bogotá, Colombia. afmoyar@unal.edu.co

1. FUNCIONES GENERALES

bool checkError(int number,int errorValue, char *message): Imprime por consola el contenido del buffer "message" y detiene la ejecución del programa si "number" es igual a "errorValue".

char *strlwr(char *str): Modifica cada caracter de la cadena "str" de tal forma que la cadena que retorna tiene todas sus letras en minúscula.

char* toLowerCase(char *str,int size): Devuelve una copia en minúsculas de la cadena de texto que se pasa como argumento.

int keypress(unsigned char echo): Espera que una tecla sea pulsada.

void toContinue(): Hace uso de la función keypress para continuar la ejecución del programa.

2. FUNCIONES DE LA TABLA HASH

void fillArray(long *array,int size,int value): Llena cada posición del arreglo al que apunta "array" con "value".

int hashFunction(unsigned char *str): Función hash que usa cada caracter de "char" para calcular y retornar un valor entero no mayor al tamaño de la tabla hash.

void initTable(long *table): Si el archivo hashTable no existe, lo crea y lo llena con -1. Si el archivo hashTable existe, lo lee y guarda sus contenidos en el buffer al que apunta "table".

void updateTable(long *table): Sobrescribe los contenidos del buffer al que apunta "table" y los escribe en el archivo hashTable.

3. FUNCIONES SOBRE LAS MASCOTAS

long getNumDogs(): Abre el archivo dataDogs para contar y retornar el número de registros en él.

int getCurrId(): Abre el archivo dataDogs para acceder a la información del último registro añadido y retornar su Id.

int recursiveCheckDog(int id, int first, int last,struct dogType *dog,FILE *file): Función recursiva auxiliar que aplica un algoritmo de búsqueda binaria sobre el archivo al que apunta file, para verificar si existe un registro con identificación "id". Los valores first y last son variables de control útiles en la recursión para acotar la zona de búsqueda; en "dog" se almacena temporalmente cada uno de los perros chequeados por la función. No usar en el programa. Esta función solo es usada por la función checkDog(int id).

int checkDog(int id): Llama la función recursiveCheckDog para averiguar si el perro identificado con "id" existe en el archivo dataDogs. De existir, retorna la dirección del registro en el archivo, de lo contrario retorna -1.

struct dogType* getDogById(int id): Usa la función checkDog para acceder a la dirección en el archivo del perro identificado por "id", y con esta accede al archivo dataDogs para leer y retornar el perro deseado. Si el perro no existe retorna un registro falso con id -1.

void overrideDog(FILE *file,long address, struct dogType *dog): Abre el archivo identificado por "file", accede a la dirección especificada por "address", que debe ser la dirección donde empieza un registro, y lo sobrescribe con la información que contiene "dog".

void sameNameUpdate(long *table, int hashIndex, long newDogAddr): Cuando se agregue un nuevo perro en la dirección de archivo "newDogAddr", esta función encuentra el último registro en dataDogs con el mismo nombre del nuevo perro y modifica el valor de su atributo next por "newDogAddr".
El parámetro "table" apunta al arreglo donde se almacena la tabla hash, y el parámetro "hashIndex" es el valor resultante de aplicar la función hash al nombre del nuevo perro. Estos parámetros se usan para encontrar la dirección del archivo del primer perro con dicho nombre.

void getListOfDogs(long *table, int hashIndex,char *lowerName,int clientfd): Accede a la posición "hashIndex" en el arreglo al que apunta "table" para obtener la dirección en el archivo del primer perro del nombre relacionado a hashIndex. Itera el archivo dataDogs para acceder a todos los registros con el mismo nombre. Envía el nombre y la dirección de cada uno de ellos al cliente identificado por "clientfd".

4. FUNCIONES DE LOS CASOS

struct dogType* enterAnimal(): Crea una estructura para que el usuario llene campo por campo los datos que se le solicitan. Los campos id y next están cargados con -1 por defecto.

void openEditor(int id,struct dogType *dog): Crea una dirección para el archivo de la mascota que se solicitó al hacer uso de su id como nombre del archivo, si no existe, lo crea o, en su defecto, lo abre en el editor de texto para que el usuario pueda ver la información básica de la mascota y pueda hacer anotaciones.

int deleteRegister(int id,long *table, int clientfd): Abre el archivo dataDogs.dat y un archivo temp.dat para copiar los datos del primero al segundo evadiendo el registro que coincide con el id que el usuario solicitó eliminar. Una vez se completa la tarea, elimina el archivo dataDogs.dat original y renombra el archivo tempo con el mismo nombre para seguir teniendo acceso a la información. Adicionalmente actualiza la tabla hash para que no se pierda la información de los animales que le siguen al que fue eliminado.

5. FUNCIONES DE LOG

void makeLog(char operation[32],struct in_addr ip,char searchedString[32]): Crea un registro de log en el archivo serverDogs.log con la siguiente información: fecha, dirección ip del cliente que solicita la opción, la opción solicitada y el nombre o id de la mascota según corresponda. Los argumentos corresponden a los últimos 3 argumentos, esta función se encarga de tomar el tiempo local que tiene el ordenador en el cual se está ejecutando el servidor. Esta función se llama en todas las opciones del menú presentado al usuario.

void readLog(): Lee el primer registro de log que exista.

6. FUNCIÓN DE HILOS

void *thread(void *ap): Esta función recibe como argumento un puntero que apunta a una estructura de tipo Argument. Cada instancia de este tipo de estructura contiene un apuntador al arreglo donde se almacena la tabla hash, un valor entero que identifica

al cliente, y una instancia de la estructura sockaddr_in que contiene la dirección IP del cliente. Esta función sirve para ser pasada como parámetro al momento de crear un hilo.

En esta función se llevan a cabo todas las opciones del servidor puesto que este crea un hilo por cada cliente que llega. Recibe la opción deseada por medio del cliente y decide cual acción llevar a cabo.