Auxiliar 6 - Archivos

Profesor: Luis Mateu
Auxiliares: Gerard Cathalifaud
Vicente González
Joaquín López
Rodrigo Urrea

Resumen

Funciones de Archivos

Para poder utilizar las funciones que manejan archivos, es necesario incluir la librería: `stdio.h`.

- `FILE *fopen(const char *filename, const char *mode)`: Abre el archivo entregado en *filename* y entrega un puntero al archivo, el *mode* corresponde al modo de acceso del archivo:
  - "r": Abre el archivo solo para lectura, el archivo debe existir.
  - "w": Crea un archivo para escribir, si existía, lo borra y comienza desde 0.
  - "a": Extiende un archivo, escribiendo justo al final, y crea si no existe.
  - "r+": Abre un archivo para leer y escribir, debe existir el archivo.
  - "w+": Crea un archivo para leer y escribir.
  - "a+": Abre un archivo para leer y extenderlo.

  Si hay un error, se retorna NULL y se activa errno.
- `int fgetc(FILE *stream)`: Lee un caracter y avanza uno en el cursor
- `char *fgets(char *s, int size, FILE *stream);` lee una linea de a lo mas $size - 1$ caracteres del stream a el buffer s o hasta que termine la linea ($\backslash n$). **OJO**: como dice el nombre, obtiene un string, entonces lo que entrega en s es terminado con el caracter nulo.
- `size_t fread(void *buf, size_t len, size_t size, FILE *stream)`: lee una cantidad de len de elementos del stream, de tamaño size a el buffer buf.
- `int fputc(char c, FILE *f)`: Escribe un caracter c en la posición del cursor actual del cursor y avanza 1
- `int fputs(const char *s, FILE *stream)`: Escribe un string en la posicion del cursor actual y avanza el tamaño del string (sin contar el nulo)
- `size_t fwrite(const void *buf, size_t len, size_t size, FILE *stream)`: Escribe una cantidad de len elementos del buf, de tamaño size a el stream (y mueve el cursor la cantidad de elementos que escribió)

- `int fclose(FILE *stream);` Cierra el archivo stream, retornando 0. En otro caso sale un error.
- `int feof(FILE *stream);` Devuelve un número distinto de 0 cuando el cursor en el archivo no está en End-Of-File.
- `int ferror(FILE *stream);` Revisa si hay un error en el archivo entregado (Error tratando de leer cuando abrió el archivo en modo lectura, o similar).
- `int fseek(FILE *stream, long int offset, int whence);` Desplaza el cursor del archivo *Stream*, una distancia *offset* desde la posición *whence*, que puede ser:
  - SEEK_SET: Desde el inicio del archivo.
  - SEEK_CUR: Desde la posición actual del cursor.
  - SEEK_END: Desde el final del archivo.
- Existe los "archivos" ya declarados, que corresponden a los estandar: *stdin, stdout, stderr*. Donde los comandos printf() y scanf() usan por default estos archivos, pero está la versión generalizada:
- `int fprintf(FILE *stream, const char *format, ...);` La versión generalizada de printf, permite enviar un texto formateado, utilizando las opciones $\backslash d$, $\backslash s$, etc. A cualquier tipo de archivo.
- `int fscanf(FILE *stream, const char *format, ...);` De igual manera, scanf corresponde a una lectura formateada, donde puede ser cualquier archivo.

Problemas

P1. (P1 C2 otoño 2016) Un archivo contiene un diccionario en el siguiente formato:

```
casa:edificación construida para ser habitada:
lluvia:condensación del vapor de agua contenida en las nubes:
embarcación:todo tipo de artilugio capaz de navegar sobre o bajo el agua:
alimento:sustancia ingerida por un ser vivo:
...etc...
```

El primer carácter ":" separa la palabra de su definición. El segundo ":" termina la definición. Todas las líneas del archivo contienen un número fijo de caracteres para que sea sencillo hacer acceso directo con `fseek`. Las palabras están desordenadas en el archivo. Este archivo no cabe en la memoria del computador. Programe la siguiente función:

```
void modificar(char *nom_dic, char *palabra, char *def, int n_lin, int ancho);
```

Esta función cambia la definición de `palabra` por `def` en el diccionario almacenado en el archivo `nom_dic`. El parámetro `n_lin` es el número de líneas del archivo (y, por lo tanto, el número de palabras y definiciones) y `ancho` es el número de caracteres de cada línea en el archivo.

Bonus: Ahora las líneas tienen un largo variable, y ya no está el ":" del final de cada línea. Tampoco se conoce la cantidad de líneas del archivo, y se agrega la condición de que si no se encuentra la palabra, entonces se debe agregar al final. Programe la función:

```
void modificar(char *nom_dic, char *palabra, char *def);
```

P2. Quicksort Cola genérica Recordando la clase auxiliar 4 de Estructura y Memoria, se desarrolló una estructura Cola para guardar string o enteros, ahora se tiene una implementación de cola genérica:

```
Queue *makeQueue();
void destroyQueue(Queue *q);
void put(Queue *q, void *ptr);
void *get(Queue *q);
void *peek(Queue *q);
int emptyQueue(Queue *q);
int queueLength(Queue *q);
```

Programe un algoritmo para ordenar los elementos de la cola genérica:

```
void quicksort(Queue *q, int (*cmp)(void *ptr1, void *ptr2));
```

Y pruebela para el caso de que los elementos correspondan a strings.

Bonus: Cree la estructura Personas, que contiene los datos, nombre (string) y edad (int), ordene una cola de Personas por la edad de manera decreciente.