



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Alejandro Esteban Pimentel Alarcón

Asignatura: Fundamentos de Programación

Grupo: 135

No de Práctica(s): 13

Integrante(s): Vanesa Bazaldúa Morales
Torres Alcántara Alan Eliezer

*No. de Equipo de
cómputo empleado:* Somalia 42

No. de Lista o 6824 y 8166

Semestre: 2020 - 1

Fecha de entrega: 11/11/2019

Observaciones:

CALIFICACIÓN: _____

Lectura y escritura de datos

Alejandro Pimentel
profpimentel9@gmail.com

Facultad de Ingeniería
UNAM
<https://qr.go.page.link/aeaVL>



4 / 4

Objetivo

Elaborar programas en lenguaje C que requieran el uso de archivos de texto plano en la resolución de problemas, entendiendo a los archivos como un elemento de almacenamiento secundario.

INTRODUCCIÓN

C dispone de una colección de funciones de biblioteca para la entrada/lectura y salida/escritura entre el ordenador y los periféricos Entrada/Salida. Se encuentran declaradas en la librería del sistema `stdio.h`, para ello se coloca en la cabecera de los programas la directiva `#include`, se considera como unidad de entrada/lectura el teclado y como unidad de salida/escritura la pantalla del ordenador.

Debido a la naturaleza dinámica de los “string”, no hay que preocuparse de la cantidad de memoria que hay que reservar para el “string”. Simplemente hay que añadir cosas y el “string” irá expandiéndose para dar cabida a lo que le introduzca.

Una de las cosas agradables de poner el fichero entero en una cadena es que la clase “string” proporciona funciones para la búsqueda y manipulación que le permiten modificar el fichero como si fuera una simple línea. Sin embargo, tiene sus limitaciones. Por un lado, a menudo, es conveniente tratar un fichero como una colección de líneas en vez de un gran bloque de texto. Por ejemplo, si quiere añadir numeración de líneas es mucho más fácil si tiene un objeto “string” distinto para cada línea. Para realizarlo, necesitamos otro concepto.

Archivos

Para trabajar con archivos en C, es necesario tener un apuntador hacia un archivo:

```
FILE *archivo;
```

Para asignar el apuntador a su lugar correspondiente, podemos contar con una función para abrir el archivo por nombre:

```
archivo = fopen("archivo.txt", "r");
```

A partir de este punto, ya podemos utilizar nuestro apuntador de archivo. Pero para leer, necesitaremos una variable en dónde guardar el texto:

```
char linea[90];  
char palabra[9];
```

Formas para abrir

Al momento de abrir un archivo (*fopen*) se puede elegir una entre varias opciones:

r: Abre un archivo de texto para lectura.
w: Crea un archivo de texto para escritura.
a: Abre un archivo de texto para añadir.
r+: Abre un archivo de texto para lectura / escritura.
w+: Crea un archivo de texto para lectura / escritura.
a+: Añade o crea un archivo de texto para lectura / escritura.

string.h

String es una librería que será de utilidad siempre que tengamos que manejar texto (cadenas de caracteres).

```
#include <string.h>
```

```
strlen(char texto[]);  
strcpy(char destino[], char origen[]);  
strcmp(char texto1[], char texto2[]);  
strcat(char destino[], char origen[]);  
strstr(char texto[], char buscado[]);  
strchr(char texto[], char buscado);
```

El secreto de las cadenas de caracteres es que indican su final con el caracter nulo "`\0`". Pero deben recordar que ese caracter también ocupa un lugar en el arreglo.

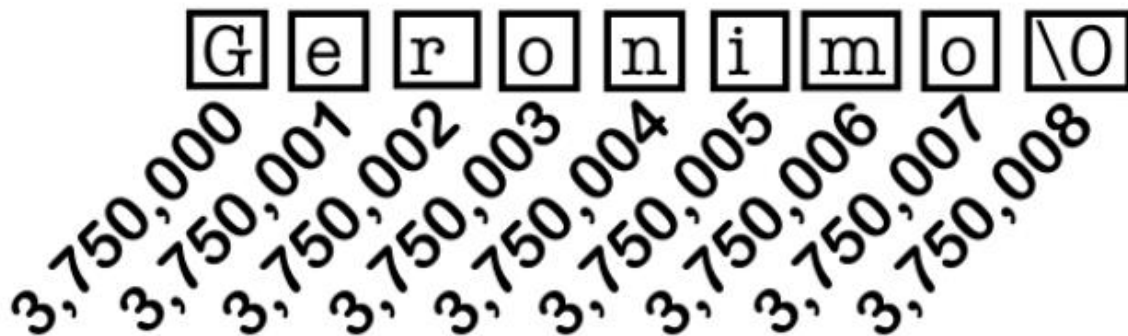


Diagrama que muestra la representación de una cadena de caracteres en memoria. La cadena es "Gerónimo\0". Los caracteres están almacenados en un arreglo de bytes, cada uno en su propia caja. Debajo de cada caja, se muestra su dirección de memoria: 3,750,000 para 'G', 3,750,001 para 'e', 3,750,002 para 'r', 3,750,003 para 'o', 3,750,004 para 'n', 3,750,005 para 'i', 3,750,006 para 'm', 3,750,007 para 'o', y 3,750,008 para el caracter nulo '\0'.

Leer

Tenemos dos formas sencillas de leer texto desde un archivo:

```
fscanf(archivo,"%8s",palabra);
```

Que funciona igual que `scanf()` con la diferencia de que como primer parámetro recibe el apuntador hacia el archivo. Recuerden que esto solo lee una palabra a la vez. El número después del porcentaje, es para limitar la cantidad de caracteres máximos que toma, esto es útil para no sobrepasar la longitud del arreglo de caracteres.

Y la otra manera es:

```
fgets(linea,89,archivo);
```

Que lee una línea completa, con un número máximo de caracteres que recibe como segundo argumento.

Observen que el orden del apuntador y la variable se invierten.

Leer

Ambas formas irán haciendo que el archivo "avance". Eso quiere decir que si las usan repetidas veces, irán leyendo nuevas palabras/líneas del archivo.

```
while( ! feof(archivo) ){  
    fgets(linea,89,archivo);  
    printf("%s",linea);  
}
```

La función *feof* nos ayuda a darnos cuenta si ya llegamos al final del archivo. La función recibe como único argumento el apuntador al archivo, y devuelve 0 mientras no sea el final del archivo.

Escribir

También hay varias formas de escribir en un archivo, pero por familiaridad, la más sencilla es *fprintf*.

```
fprintf(archivo,"%sn","texto");
```

Al igual que con *fscanf*, lo único que cambia es que el primer parámetro es el apuntador del archivo.

Por último, hay que cerrar el archivo que abrimos con *fopen*. Se cierra con *fclose*.

`fclose(archivo);`

Actividad

Crear un programa que pida el nombre de un archivo de entrada y un archivo de salida.

Para el archivo de entrada, mostrar:

- Texto.
- Número de líneas.
- Número de palabras (cualquier cosa entre espacios).
- Número de caracteres.

Para el archivo de salida:

- Copiar el archivo de entrada con las líneas invertidas.


```

1  #include<stdio.h>
2  #include<string.h>
3  int main(){
4
5      FILE *archivo, *archivosalida;
6
7      char palabra [20], linea[100];
8      printf("Nombre del archivo\n");
9      char nombre[30];
10     scanf("%s",nombre);
11     char nombresalida[30];
12     printf("Nombre del archivo nuevo\n");
13     scanf("%s",nombresalida);
14
15     archivo=fopen(nombre,"r");
16     int contadorlineas=0;
17
18     while(!feof(archivo)){
19         fgets(linea,100,archivo);
20         printf("%s",linea);
21         contadorlineas++;
22     }
23

```

Incluimos la librería string para poder usar algunas funciones especiales que nos permitan trabajar dentro de los archivos que deseamos, usamos dos apuntadores para los archivos, colocamos unas variables para poder guardar texto y utilizamos scanf para que el usuario pudiese colocar cualquier archivo. Después mandamos la función para que lo abra, y colocamos un ciclo para recibir el número de líneas.

```

24     printf("\n numero de lineas: %i\n",contadorlineas);
25     int contadorpalabras=0;
26
27     archivo=fopen(nombre,"r");
28     while(!feof(archivo)){
29         fscanf(archivo,"%8s",palabra);
30         contadorpalabras++;
31     }
32
33     printf("\n numero de palabras: %i\n",contadorpalabras);
34     archivo=fopen(nombre,"r");
35     int contadorcaracteres=0,npalabra;
36
37     while(!feof(archivo)){
38         fscanf(archivo,"%8s",palabra);
39         npalabra=strlen(palabra);
40         contadorcaracteres=contadorcaracteres+npalabra;
41     }
42     printf("\n numero de caracteres: %i\n",contadorcaracteres);
43     archivo=fopen(nombre,"r");
44     archivosalida=fopen(nombresalida,"w");
45
46     char listaarchivo[contadorlineas][100];
47

```


Después repetimos el mismo proceso para poder leer el número de palabras y al final el número de caracteres, la única diferencia se encuentra en la función strlen que nos sirve para leer caracteres únicamente.

```
47
48     for(int i=contadorlineas-1;i!=-1;i--){
49         fgets(listaarchivo[i],100,archivo);
50     }
51
52     for(int i=0;i<contadorlineas;i++){
53         if(i==0){
54             fprintf(archivosalida,"%s\n",listaarchivo[i]);
55         }
56         else{
57             fprintf(archivosalida,"%s",listaarchivo[i]);
58         }
59     }
60     return 0;
61
62 }
```

Al final, en el cierre del programa usamos dos for para invertir las líneas del texto de atrás hacia adelante, pero se usaron para retroceder sin invertir los caracteres, solo las líneas, al final se manda imprimir el archivo en el orden que deseamos.

```
Nombre del archivo
calaverita.txt
Nombre del archivo nuevo
alreves.txt
?1??
```

Corrimos el programa y nos pidió dos archivos.

```
numero de lineas: 57
numero de palabras: 199
numero de caracteres: 855
```

Nos mostró el número de líneas, palabras y caracteres.

Autora: Davina Gpe. Ponce Mtz.

Y esperamos su regreso!
Recordamos sus amores,
De saber que las queremos,
Queridas almas contentas,

Con atoles y galletas.
Estas dos noches completas
Pues sé que departiremos
Yo los espero sentada

Y se harían muy amenos!
Vendrían todos los días
Cuánto los echo de menos,
Si supieran mis muertitos

Y poderlos disfrutar.
En fechas tan especiales
El poderlos encontrar,
Para mí es un gran regalo

Los recuerdos que se van.
En esta tierra de amores
En su visita fugaz,
Para que pasen contentos

Se los hemos de dejar.
Con formas de cuerno y hueso
Que de Colores están,
Estos panes primorosos,

Con flores velas, mezcal!
Colocados en altares
Con grandes piezas de pan,
Así que los festejamos

Los sentimos regresar!
Pues sus almas y latidos
Les ponemos un altar,
A nuestros seres queridos

Recordamos sus andanzas.
De volvernos a encontrar,
Que aún en la confianza,
Es menester recordar

Tradiciones y alabanzas.
Recordamos con amor
En que todo es fiesta y danza,
En este mes singular,
?1??
?1??

Así fue nuestro resultado del programa al ejecutarlo, primero nos mostro el numero de palabras, caracteres y líneas, después mostró el texto original escrito al revés

CONCLUSIÓN:

Los archivos de texto plano para la resolución de problemas son de vital importancia, aprenderlos en C es importante porque así podemos editar textos planos, leerlos, y recibir datos de importancia e interpretarlos como sea nuestra conveniencia desde un programa en C, es una excelente herramienta para leer dentro de códigos fuentes ya que la mayoría de éstos se escriben en texto plano.

Con la librería string podemos leer dentro del archivo que abrimos en C, también es importante guardar el texto que ocupamos en el texto en otra variable para poder utilizarlo después dependiendo del uso que le queramos dar, lo importante al momento de usar estos programas es que al escribirlos usamos el mismo lenguaje en C con "if", "for", "while", "do while", etc. Gracias a los fundamentos que aprendimos podemos utilizar este tipo de herramientas de edición de archivos.