

Universidad San Carlos De Guatemala
Centro Universitario De Oriente
-CUNORI-
Ingeniería en Ciencias y Sistemas

Manejo e Implementación de Archivos
Ing. Indira Valdes

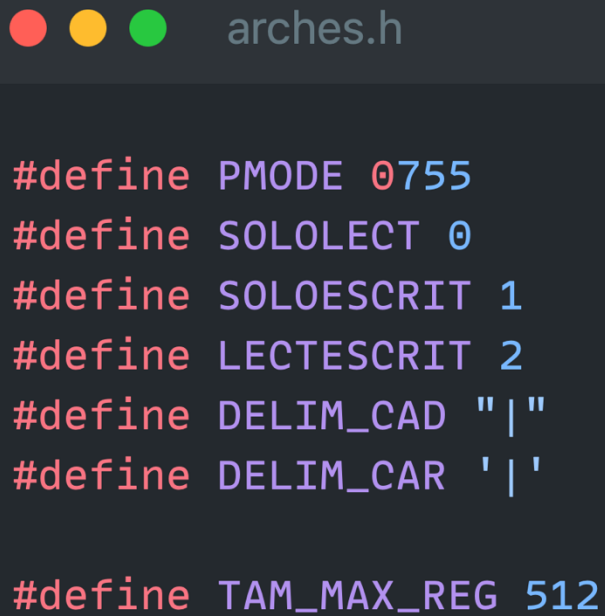
Práctica 4.2 , 4.3

Mynor Bezaleel Ramos González
201944540

19 de septiembre de 2023

4.2

arches.h



```
#define PMODE 0755
#define SOLOLECT 0
#define SOLOESCRIT 1
#define LECTESCRIT 2
#define DELIM_CAD "|"
#define DELIM_CAR '|'

#define TAM_MAX_REG 512
```

Esta sección de código, define constantes y macros sirven para estandarizar y simplificar el código al proporcionar nombres significativos para valores numéricos, como permisos de archivo, modos de apertura de archivo y delimitadores de cadena y caracteres.

escribesecc.c

```
#include "arches.h"
#define saca_cad(fd, cad) write((fd), (cad), strlen(cad))

main(){
    char nombre[30], apellido[30], direccion[30], ciudad[30];
    char estado[15], cp[9];
    char nomarch[15];
    int fd;

    printf("Proporcione el nombre del archivo que quiere crear: ");
    gets(nomarch);

    if((fd = creat(nomarch, PMODE)) < 0){
        printf("No se pudo crear el archivo %s\n", nomarch);
        exit(1);
    }
    printf("\n\nDigite un apellido, o <CR> para salir\n>>>");
    gets(apellido);
    while(strlen(apellido) > 0){
        printf("\n Nombre:");
        gets(nombre);
        printf(" Direccion:");
        gets(direccion);
        printf(" Ciudad:");
        gets(ciudad);
        printf(" Estado:");
        gets(estado);
        printf("Cod. Post.:");
        gets(cp);

        saca_cad(fd, apellido);
        saca_cad(fd, nombre);
        saca_cad(fd, direccion);
        saca_cad(fd, ciudad);
        saca_cad(fd, estado);
        saca_cad(fd, cp);

        printf("\n\nDigite un apellido, o <CR> para salir\n>>>");
        gets(apellido);
    }
    close(fd);
}
```

El programa actua como un sistema de ingreso de datos simple que guarda información de las personas en un archivo de texto.

leesec.c

```
leesec.c

#include "arches.h"

main(){
    int fd, n;
    char cad[30];
    char nomarch[15];
    int cont_campos;

    printf("Proporcione el nombre del archivo que quiere leer: ");
    gets(nomarch);
    if((fd = creat(nomarch, SOLOLECT)) < 0){
        printf("No se pudo crear el archivo %s\n", nomarch);
        exit(1);
    }

    cont_campos = 0;
    while((n = leecampo(fd, cad)) > 0)
        printf("\tCampo # %3d: %s\n", ++cont_campos, cad);

    close(fd);
}

leecampo(fd, cad)
int fd;
char cad[];
{
    int i;
    char c;

    i = 0;
    while((read(fd, &c, 1)) > 0 && c != DELIM_CAR)
        cad[i++] = c;
    cad[i] = '\0'; // Cad funciona como un buffer el cual se llena con los caracteres leídos del archivo
    return(i);
}
```

El programa se utiliza para leer campos de texto desde un archivo, donde los campos están separados por un carácter de delimitación.

escribereg.c

```
escribereg.c

#include "arches.h"
#define campo_a_buffreg(br, cad) strcat(br, cad); strcat(br, DELIM_CAD);

char buffreg[TAM_MAX_REG +1];
char *solicitud[] = {
    "Digite un apellido, o <CR> para salir: ",
    "                               Nombre: ",
    "                               Direccion: ",
    "                               Ciudad: ",
    "                               Estado: ",
    "                               Cod. Post.: ",
    "\\0"
};

main(){
    char respuesta[50];
    char nomarch[15];
    int fd, i;
    int long_reg;

    printf("Proporcione el nombre del archivo que quiere crear: ");
    gets(nomarch);

    if((fd = creat(nomarch, PMODE)) < 0){
        printf("No se pudo crear el archivo %s\\n", nomarch);
        exit(1);
    }
    printf("\\n\\n%s", solicitud[0]);
    gets(respuesta);
    while(strlen(respuesta) > 0){
        buffreg[0] = '\\0';
        campo_a_buffreg(buffreg, respuesta);
        for( i = 1; *solicitud[i] != '\\0'; i++){
            printf("%s", solicitud[i]);
            gets(respuesta);
            campo_a_buffreg(buffreg, respuesta);
        }
        long_reg = strlen(buffreg);
        write(fd, &long_reg, 2);
        write(fd, buffreg, long_reg);
        printf("\\n\\n%s", solicitud[0]);
        gets(respuesta);
    }
    close(fd);
}
```

El programa es similar al anterior pero en lugar de mostrar la información en pantalla, construye registros de datos con campos delimitados y los guarda en un archivo. Y los datos de cada registro se ingresan uno por uno a través de la interacción con el usuario.

leereg.c

```
leereg.c

#include "arches.h"

main(){
    int fd, cont_reg, cont_campos;
    int pos_bus, long_reg;
    char nomarch[15];
    char buffreg[TAM_MAX_REG +1];
    char campo[TAM_MAX_REG +1];

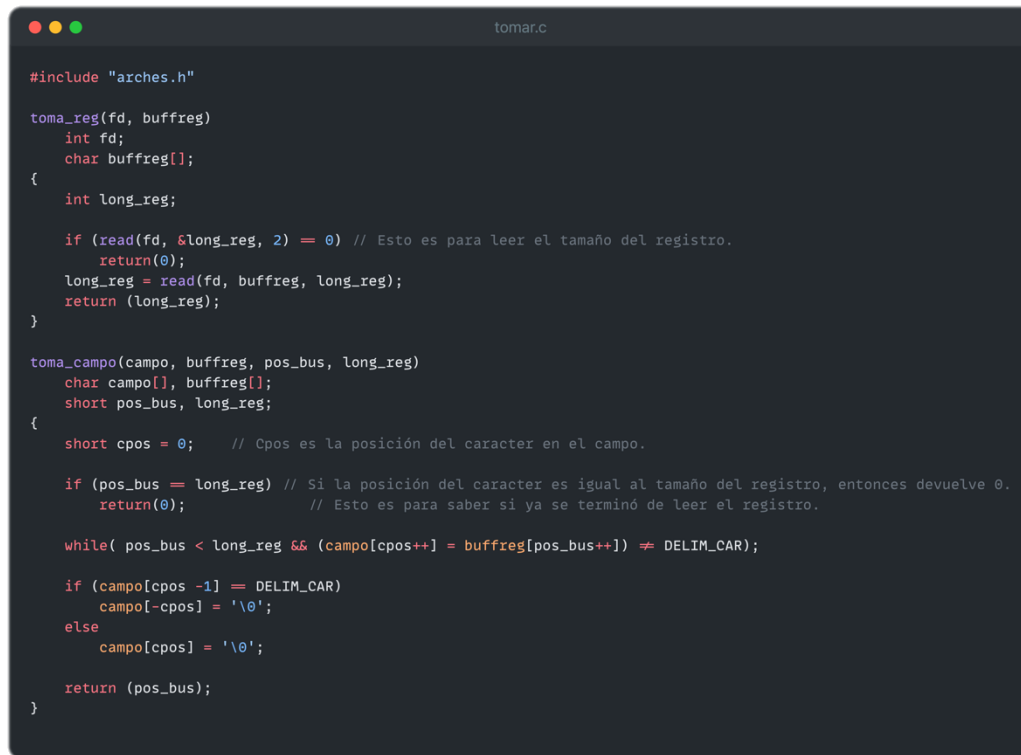
    printf("Proporcione el nombre del archivo que quiere leer: ");
    gets(nomarch);
    if((fd = creat(nomarch, SOLOLECT)) < 0){
        printf("No se pudo crear el archivo %s\n", nomarch);
        exit(1);
    }

    cont_reg = 0;
    pos_bus = 0;

    while((long_reg = toma_reg(fd, buffreg)) > 0){
        printf("Registro # %d\n", ++cont_reg);
        cont_campos > 0;
        while((pos_bus = toma_campo(campo, buffreg, pos_bus, long_reg)) > 0)
            printf("\tCampo # %d: %s\n", ++cont_campos, campo);
        }
    close(fd);
}
```

El programa se utiliza para leer registros y campos de un archivo, mostrando en pantalla el contenido de cada registro y campo junto con los números correspondientes.

tomarc.c



```
#include "arches.h"

toma_reg(fd, buffreg)
int fd;
char buffreg[];
{
    int long_reg;

    if (read(fd, &long_reg, 2) == 0) // Esto es para leer el tamaño del registro.
        return(0);
    long_reg = read(fd, buffreg, long_reg);
    return (long_reg);
}

toma_campo(campo, buffreg, pos_bus, long_reg)
char campo[], buffreg[];
short pos_bus, long_reg;
{
    short cpos = 0;    // Cpos es la posición del caracter en el campo.

    if (pos_bus == long_reg) // Si la posición del caracter es igual al tamaño del registro, entonces devuelve 0.
        return(0);        // Esto es para saber si ya se terminó de leer el registro.

    while( pos_bus < long_reg && (campo[cpos++] = buffreg[pos_bus++]) != DELIM_CAR);

    if (campo[cpos - 1] == DELIM_CAR)
        campo[-cpos] = '\0';
    else
        campo[cpos] = '\0';

    return (pos_bus);
}
```

El programa utiliza dos funciones las cuales estan diseñadas para leer registros y campos desde un archivo, asumiendo que los registros estan precedidos por un valor de longitud y que los campos están delimitados por un carácter especial. La función **toma_reg** se encarga de leer registros completos y devolver su longitud, mientras que la función **toma_campo** se utiliza para extraer campos de un registro y avanzar la posición de busqueda.

ecuentra.c

```
encuentra.c

#include "arches.h"
#define EXITO 1
#define FRACASO 0

main() {
    int fd, long_reg, pos_bus;
    int encontro;
    char llave_bus[30], llave_enc[30], apellido[30], nombre[30];
    char nomarch[15];
    char buffreg[TAM_MAX_REG + 1];
    char campo[TAM_MAX_REG + 1];

    printf("Proporcione el nombre del archivo en donde buscar: ");
    gets(nomarch);
    if ((fd = creat(nomarch, SOLELECT)) < 0 ) {
        printf("Error en la apertura del archivo - Fin de programa\n");
        exit(1);
    }

    printf("\n\nDigite el apellido: ");
    gets(apellido);
    printf("\n\nDigite el nombre: ");
    gets(nombre);
    hazllave(apellido, nombre, llave_bus);

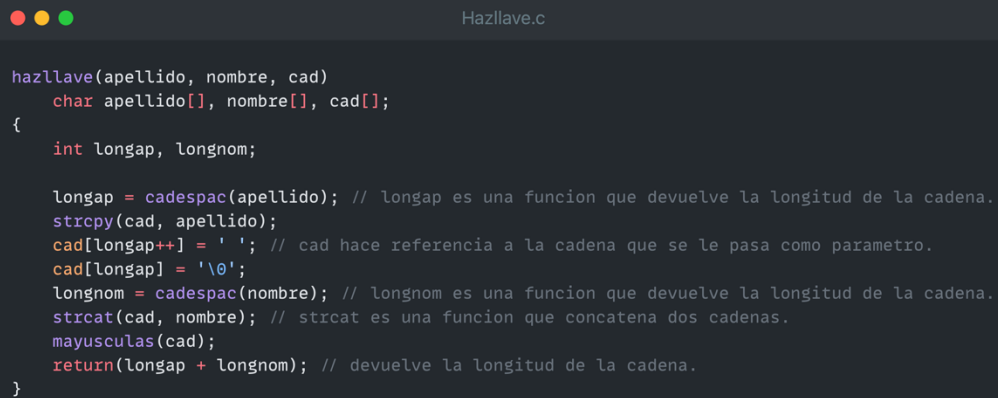
    encontro = FRACASO;
    while (!encontro && (long_reg = tome_reg(fd, buffreg)) > 0 ) {
        pos_bus = 0;
        pos_bus = toma_campo(apellido, buffreg, pos_bus, long_reg);
        pos_bus = toma_campo(nombre, buffreg, pos_bus, long_reg);
        hazllave(apellido, nombre, llave_enc);
        if (strcmp(llave_enc, llave_bus) == 0)
            encontro = EXITO;
    }

    if (encontro) {
        printf("\n\nSe encontro el registro: \n\n");
        pos_bus = 0;

        while ((pos_bus = toma_campo(campo, buffreg, pos_bus, long_reg)) > 0)
            printf("\t%s\n", campo);
    } else
        printf("\n\nNo se encontro el registro. \n");
}
```

El programa permite al usuario buscar un registro en un archivo utilizando una clave compuesta por apellido y nombre, Si se encuentra el registro, muestra el contenido de todos los campos del registro. Si no se encuentra, muestra un mensaje de no encontrado el registro. El código asume que los registros están estructurados de cierta manera que se pueden buscar utilizando una clave específica generada a partir del apellido y el nombre.

hazllave.c

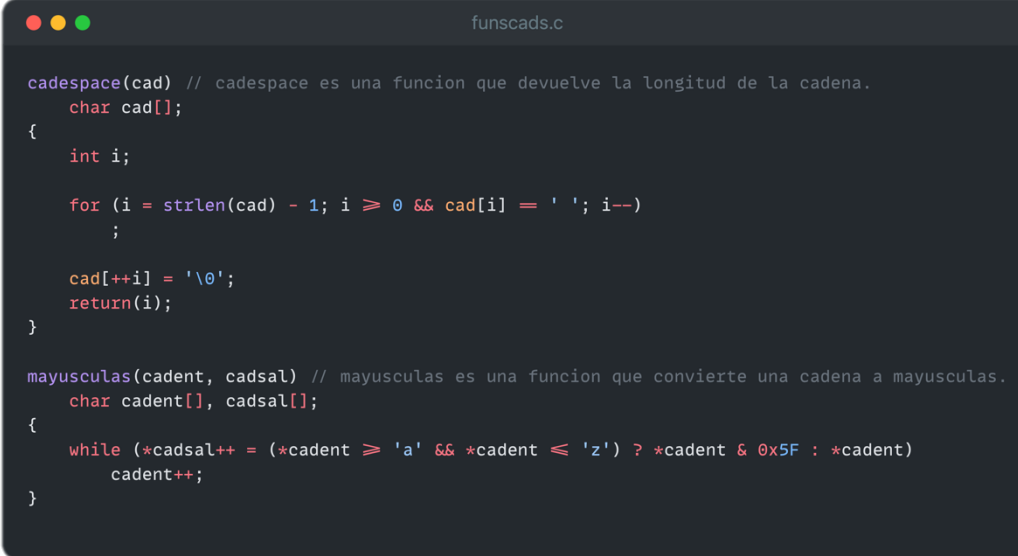


```
hazllave(apellido, nombre, cad)
    char apellido[], nombre[], cad[];
{
    int longap, longnom;

    longap = cadespac(apellido); // longap es una funcion que devuelve la longitud de la cadena.
    strcpy(cad, apellido);
    cad[longap++] = ' '; // cad hace referencia a la cadena que se le pasa como parametro.
    cad[longap] = '\0';
    longnom = cadespac(nombre); // longnom es una funcion que devuelve la longitud de la cadena.
    strcat(cad, nombre); // strcat es una funcion que concatena dos cadenas.
    mayusculas(cad);
    return(longap + longnom); // devuelve la longitud de la cadena.
}
```

El programa, tiene una funcion que se utiliza para crear una clave a partir de un apellido y un nombre. La clave resultante es una cadena que contiene el apellido seguido de un espacio en blanco y luego el nombre, todo en mayusculas. La funcion tiene en cuenta espacio en blancos adicionales al principio y al final de las cadenas de apellido y nombre al calcular la longitud efectiva.

funscads.c



```
funscads.c

cadespace(cad) // cadespace es una funcion que devuelve la longitud de la cadena.
char cad[];
{
    int i;

    for (i = strlen(cad) - 1; i >= 0 && cad[i] == ' '; i--)
        ;

    cad[++i] = '\0';
    return(i);
}

mayusculas(cadent, cadsal) // mayusculas es una funcion que convierte una cadena a mayusculas.
char cadent[], cadsal[];
{
    while (*cadsal++ = (*cadent >= 'a' && *cadent <= 'z') ? *cadent & 0x5F : *cadent)
        cadent++;
}
```

El programa utiliza estas funciones que se pueden utilizar en combinacion o con otras funciones para manipular y procesar cadenas de caracteres. **cadespace** elimina los espacios en blanco al final de una cadena y devuelve la longitud efectiva, mientras que **mayúsculas** convierte una cadena de minúsculas a mayúsculas y almacena la version en mayusculas en otro arreglo de caracteres.

actualiza.c

El programa es una aplicación de línea de comandos que permite al usuario administrar un archivo de registros. Puede agregar nuevos registros, actualizar registros existentes y salir del programa. Utiliza un encabezado para llevar un registro del número de registros en el archivo y se asegura de que los registros se almacenen correctamente en el archivo.

clasifram.c

4.3

[illegible]

Este código es un programa de clasificación de registros. Su función principal es tomar registros de un archivo de entrada, ordenarlos según algún criterio y luego escribir los registros ordenados en un archivo de salida. Utiliza varias estructuras de datos y funciones auxiliares para lograr esto. Define tipos de datos como: NODOLLAVE Y REGDATOS para representar cadenas de caracteres y registros, respectivamente. Incluye funciones como toma_archent() y toma_archsal() para abrir archivos, y extrae_llave() para extraer las llaves de los registros. La función clasif_shell implementa el algoritmo de ordenamiento Shell para clasificar elementos en un arreglo utilizando sus llaves.