

# Memoria Ejercicio Final

Autor: Martina García González NP:141478

## 1. Índice de contenidos

1.	ÍNDICE DE CONTENIDOS .....	2
2.	SOLUCIÓN .....	3
2.1.	VISTA GENERAL PROGRAMA.....	3
2.2.	GESTIÓN RUTAS DE SERVIDOR .....	3
2.3.	COPIA DE ARCHIVOS CON LLAMADAS AL SISTEMA .....	4
2.4.	FUNCIONES ASOCIADAS A HEBRAS .....	5
2.5.	INICIO DEL PROGRAMA Y PREPARACIÓN DEL SERVIDOR.....	6
2.6.	APARTADO 1- CREACIÓN DIRECTORIOS (ADDDIR) .....	7
2.7.	APARTADO 1- ELIMINACIÓN DE DIRECTORIOS (RMDIR).....	7
2.7.	APARTADO 2 – COPIA DE ARCHIVOS AL SERVIDOR (ADDARCHIVO) .....	7
2.8.	APARTADO 2 – CREACIÓN DE ARCHIVO DE CEROS (ADDARCHIVOCEROS) .....	8
2.9.	APARTADO 3 – ELIMINACIÓN DE ARCHIVOS CON HEBRAS (RMARCHIVO).....	8
2.10.	APARTADO 3 – COPIA DE ARCHIVOS CON HEBRAS (GETARCHIVO) .....	9
2.11.	PRUEBAS REALIZADAS.....	9
3.	ANEXO 1: CÓDIGO DE LA SOLUCIÓN .....	11
•	<i>Github: <a href="https://github.com/martinagg7/so_feedback.git">https://github.com/martinagg7/so_feedback.git</a></i> .....	11

## 2. Solución

### 2.1. Vista general Programa

En este programa se han definido varias funciones, cada una con una tarea concreta dentro del servidor:

- La función **ruta\_base** se encarga de **construir correctamente las rutas de archivos y directorios** dentro del servidor, asegurando que todas las operaciones se realicen dentro del directorio miServidorWebDir.
- La función **copiar\_archivo** permite **copiar archivos** utilizando llamadas al sistema, leyendo el contenido de un archivo y escribiéndolo en otro.
- La función **borrar\_archivo** permite **eliminar un archivo del servidor** y se ejecuta mediante una hebra.
- La función **get\_archivo** copia un **archivo del servidor a una ruta local** y también se ejecuta mediante una hebra.

El programa utiliza **procesos** para las siguientes operaciones:

- **addDir**, para crear directorios dentro del servidor.
- **rmDir**, para eliminar directorios del servidor.
- **addArchivo**, para copiar archivos al servidor.
- **addArchivoCeros**, para crear archivos de un tamaño determinado.

Estas operaciones se ejecutan en procesos independientes porque implican un mayor trabajo sobre el sistema. De esta forma, el proceso principal queda separado de la ejecución de estas tareas.

Por otro lado, el programa utiliza **hebras** para las siguientes operaciones:

- **rmArchivo**, para borrar archivos del servidor.
- **getArchivo**, para copiar archivos desde el servidor al sistema local.

El uso de hebras permite realizar estas operaciones de forma más ligera. Esto hace que estas acciones se ejecuten de manera más sencilla y eficiente dentro del programa.

### 2.2. Gestión Rutas de Servidor



```
void ruta_base(char *destino, const char *subruta) {
    char *home = getenv("HOME");
    sprintf(destino, "%s/%s/%s", home, BASE_DIR, subruta);
}
```

La función `ruta_base` recibe dos cosas:

- un espacio donde **guardar la ruta final** (*destino*)
- el **nombre del directorio** o archivo que se quiere usar (*subruta*)

Dentro de la función, primero se obtiene el directorio HOME del usuario.

Después, se unen tres partes:

1. el directorio HOME
2. el directorio del servidor (*miServidorWebDir*)
3. la ruta indicada en la orden

El resultado es una ruta completa dentro del servidor, que se guarda en *destino*. Esa ruta es la que se usa luego para crear, borrar o copiar archivos y directorios.

## 2.3. Copia de Archivos con Llamadas al Sistema



```
void copiar_archivo(const char *origen, const char *destino) {
    int fd_origen = open(origen, O_RDONLY);
    int fd_destino = open(destino, O_WRONLY | O_CREAT | O_TRUNC, 0644);

    char buffer[BUFFER];
    int leidos;

    while ((leidos = read(fd_origen, buffer, BUFFER)) > 0) {
        write(fd_destino, buffer, leidos);
    }

    close(fd_origen);
    close(fd_destino);
}
```

La función `copiar_archivo` recibe:

- la **ruta del archivo** que se quiere copiar (*origen*)
- la **ruta donde se quiere crear la copia** (*destino*)

Primero se abre el archivo de origen solo para leer.

Después se abre el archivo de destino para escribir, borrando su contenido si ya existía.

A continuación, el archivo se copia poco a poco:

- se leen bloques de datos del archivo de origen
- esos datos se escriben en el archivo de destino
- este proceso se repite hasta que ya no queda nada por leer

Al final, se cierran ambos archivos.

## 2.4. Funciones Asociadas a Hebras

En el apartado 3 del programa se utilizan **hebras** para realizar algunas operaciones del servidor. En lugar de crear nuevos procesos, estas operaciones se ejecutan dentro del mismo proceso principal, pero en una hebra distinta.

Las hebras se usan porque las tareas que realizan son simples y rápidas, y no es necesario crear un proceso completo para ellas

[Borrar Archivo\(\)](#)



```
void *borrar_archivo(void *arg) {
    char *ruta = (char *)arg;
    unlink(ruta);
    pthread_exit(NULL);
}
```

Esta función se ejecuta dentro de una **hebra** y se utiliza para eliminar un archivo del servidor.

- Recibe como parámetro la ruta del archivo a eliminar.
- Llama a **unlink** para **borrar el archivo**.
- Cuando termina, finaliza la hebra con **pthread\_exit**.

[GetArchivo\(\)](#)

```
● ● ●  
void *get_archivo(void *arg) {  
    char **rutas = (char **)arg;  
    copiar_archivo(rutas[0], rutas[1]);  
    pthread_exit(NULL);  
}
```

Esta función también se ejecuta dentro de una **hebra** y se usa para copiar un archivo.

- Recibe dos rutas:
  - la del archivo dentro del servidor
  - la ruta donde se copiará en el sistema local
- Utiliza la función **copiar\_archivo** para realizar la copia.
- Finaliza la hebra cuando termina la operación.

## 2.5. Inicio del programa y preparación del servidor

```
● ● ●  
  
int main(int argc, char *argv[]) {  
  
    if (argc < 3) return -1;  
  
    char base[BUFFER];  
    char ruta[BUFFER];  
  
    char *home = getenv("HOME");  
    sprintf(base, "%s/%s", home, BASE_DIR);  
    mkdir(base, 0755);
```

- Se comprueba el valor de argc para asegurarse de que el programa ha recibido al menos los argumentos necesarios.
- Se inicializan las variables **base** y **ruta**, que se utilizarán para construir las rutas del directorio principal del servidor y de los archivos .
- Se obtiene el **directorio HOME** del usuario mediante getenv("HOME")
- Con la variable **home** y la constante **BASE\_DIR**, se construye la **ruta del directorio base** del servidor en la variable **base**.
- Se crea el **directorio miServidorWebDir** utilizando mkdir en caso de que no exista previamente.

## 2.6. APARTADO 1- Creación Directorios (addDir)

```
● ● ●

if (strcmp(argv[1], "addDir") == 0) {
    pid_t pid = fork();
    if (pid == 0) {
        ruta_base(ruta, argv[2]);
        mkdir(ruta, 0755);
        exit(0);
    }
    wait(NULL);
    return 0;
}
```

- Se compara argv[1] con la cadena "addDir" para identificar la orden.
- Se crea un nuevo proceso con fork y su identificador se guarda en pid.
- Cuando pid == 0, el código se ejecuta en el proceso hijo. El **proceso hijo construye la ruta del nuevo directorio** usando ruta\_base y argv[2].
- El directorio se crea con mkdir.
- El proceso hijo finaliza con exit(0). El proceso padre llama a wait para esperar a que el hijo termine.

## 2.7. APARTADO 1- Eliminación de Directorios (rmDir)

```
● ● ●

if (strcmp(argv[1], "rmDir") == 0) {
    pid_t pid = fork();
    if (pid == 0) {
        ruta_base(ruta, argv[2]);
        rmdir(ruta);
        exit(0);
    }
    wait(NULL);
    return 0;
}
```

- Se detecta la orden comparando argv[1] con "rmDir".
- Se crea un proceso hijo mediante fork. Este construye la ruta del directorio usando ruta y argv[2].
- El directorio se elimina con rmdir.

## 2.7. APARTADO 2 – Copia de archivos al servidor (addArchivo)

```

● ● ●

if (strcmp(argv[1], "addArchivo") == 0) {
    pid_t pid = fork();
    if (pid == 0) {
        char destino[BUFFER];
        ruta_base(destino, argv[3]);
        strcat(destino, "/");
        strcat(destino, strrchr(argv[2], '/') + 1);
        copiar_archivo(argv[2], destino);
        exit(0);
    }
    return 0;
}

```

- En el proceso hijo:
  - Se declara la **variable destino** para la ruta final del archivo.
  - Se construye la **ruta del directorio destino** usando argv[3].
  - Se obtiene el **nombre del archivo** original desde argv[2].
  - Se copia el archivo usando la función copiar\_archivo.

## 2.8. APARTADO 2 – Creación de archivo de ceros (addArchivoCeros)

```

● ● ●

if (strcmp(argv[1], "addArchivoCeros") == 0) {
    pid_t pid = fork();
    if (pid == 0) {
        int tam = atoi(argv[4]);
        ruta_base(ruta, argv[3]);
        strcat(ruta, "/");
        strcat(ruta, argv[2]);
        int fd = open(ruta, O_CREAT | O_WRONLY | O_TRUNC, 0644);
        for (int i = 0; i < tam; i++) write(fd, "0", 1);
        close(fd);
        exit(0);
    }
    return 0;
}

```

- En el proceso hijo:
  - Se convierte argv[4] a entero y se guarda en tam.
  - Se construye la ruta completa del archivo usando ruta y argv[3].
  - **Se crea el archivo con open.**
  - Se escriben tam caracteres '0'.

## 2.9. APARTADO 3–Eliminación archivos con hebras (rmArchivo)

```

if (strcmp(argv[1], "rmArchivo") == 0) {
    pthread_t hebra_borrado;
    ruta_base(ruta, argv[2]);
    pthread_create(&hebra_borrado, NULL, borrar_archivo, r
    pthread_join(hebra_borrado, NULL);
    return 0;
}
return go(f, seed, [])
}

```

- Se crea la hebra **hebra\_borrado**, que será la encargada de **eliminar el archivo indicado**.
- Se construye la ruta del archivo que se quiere borrar dentro del servidor usando el valor de argv[2].
- Se lanza la hebra para que ejecute la función `borrar_archivo`, que elimina el archivo del servidor.

## 2.10. APARTADO 3 – Copia de archivos con hebras (getArchivo)

```

● ● ●

if (strcmp(argv[1], "getArchivo") == 0) {
    pthread_t hebra_copia;
    char origen[BUFFER];
    ruta_base(origen, argv[2]);
    char *rutas[2] = {origen, argv[3]};
    pthread_create(&hebra_copia, NULL, get_archivo, rutas);
    pthread_join(hebra_copia, NULL);
    return 0;
}

```

- Se crea la hebra **hebra\_copia**, que será la encargada de **copiar el archivo solicitado**.
- Se construye la ruta del archivo que está almacenado en el servidor y se guarda en la variable `origen` usando el valor de `argv[2]`.
- Se preparan las rutas de origen y destino en el array `rutas`, indicando qué archivo se copia y dónde se guarda la copia.
- **Se lanza la hebra para que ejecute la función `get_archivo`**, que realiza la copia del archivo,

## 2.11. Pruebas Realizadas

```
● (base) martinagarciagonzalez@MacBook-Air-de-Martina-2 feedback_final % clang miServidorWeb.c -o miServidorWeb -lpthread
● (base) martinagarciagonzalez@MacBook-Air-de-Martina-2 feedback_final % ./miServidorWeb addDir dirA
● (base) martinagarciagonzalez@MacBook-Air-de-Martina-2 feedback_final % ./miServidorWeb addDir dirA
./miServidorWeb rmDir dirA
● (base) martinagarciagonzalez@MacBook-Air-de-Martina-2 feedback_final % ./miServidorWeb addDir dirA
● (base) martinagarciagonzalez@MacBook-Air-de-Martina-2 feedback_final % ./miServidorWeb addArchivo /tmp/prueba.txt dirA
● (base) martinagarciagonzalez@MacBook-Air-de-Martina-2 feedback_final % ./miServidorWeb addArchivoCeros ceros.txt dirA 10
● (base) martinagarciagonzalez@MacBook-Air-de-Martina-2 feedback_final % ./miServidorWeb rmArchivo dirA/prueba.txt
./miServidorWeb getArchivo dirA/ceros.txt /tmp/copia.txt
● (base) martinagarciagonzalez@MacBook-Air-de-Martina-2 feedback_final % ls ~/miServidorWebDir/dirA
ls -l ~/miServidorWebDir/dirA
cat ~/miServidorWebDir/dirA/ceros.txt

ceros.txt
total 8
-rw-r--r-- 1 martinagarciagonzalez staff 10 Feb 6 20:39 ceros.txt
0000000008
● (base) martinagarciagonzalez@MacBook-Air-de-Martina-2 feedback_final % █
```

Para comprobar el correcto funcionamiento del servidor, se han realizado varias pruebas ejecutando las distintas órdenes implementadas desde la línea de comandos.

En primer lugar, el programa se compila correctamente utilizando el compilador clang y enlazando la librería de hebras:

- Se genera el ejecutable miServidorWeb sin errores de compilación.

A continuación, se prueban las distintas funcionalidades del servidor:

- Se crea un directorio dentro del servidor utilizando la orden **addDir**, comprobando que el directorio se crea correctamente dentro de miServidorWebDir.
  - Se elimina el directorio creado mediante la orden **rmDir**, verificando que la eliminación se realiza correctamente.
  - Se vuelve a crear el directorio y se copia un archivo desde el sistema local al servidor utilizando la orden **addArchivo**.
  - Se crea un archivo llamado ceros.txt con un tamaño de 10 caracteres utilizando la orden **addArchivoCeros**.
  - Se elimina un archivo del servidor mediante la orden **rmArchivo**.
  - Se copia un archivo desde el servidor al sistema local utilizando la orden **getArchivo**, generando una copia en la ruta indicada.

Finalmente, se comprueba el contenido del directorio del servidor utilizando comandos del sistema (ls, ls -l y cat). Se verifica que el archivo ceros.txt existe, tiene un tamaño de 10 bytes y contiene únicamente caracteres 0, lo que confirma que las operaciones se han ejecutado correctamente.

### 3. Anexo 1: Código de la solución

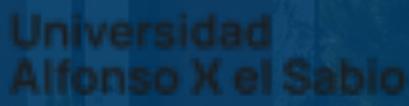
- Github: [https://github.com/martinagg7/so\\_feedback.git](https://github.com/martinagg7/so_feedback.git)



WELCOME  
TO  
**UAX**



**UAX**



Universidad  
Alfonso X el Sabio



GRACIAS