

Pontificia Universidad
JAVERIANA
Bogotá

Juan Clavijo

Jorge Torrado

Taller 01

02/09/2023

Sistemas Operativos

John Jairo Corredor

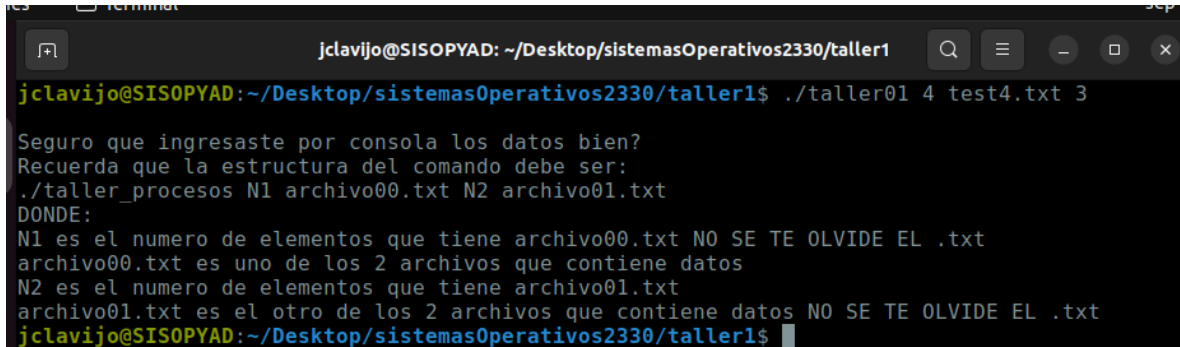
Objetivos:

- Aplicar los conceptos sobre procesos y su comunicación

Desarrollo del Taller

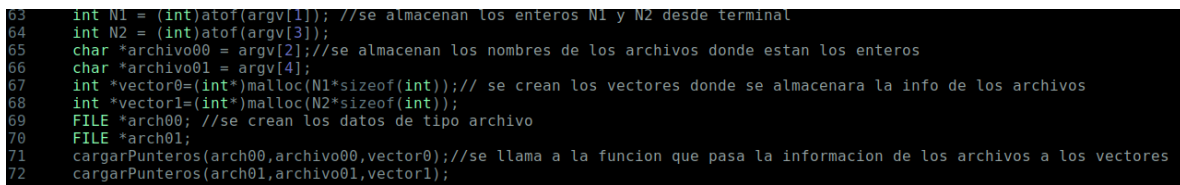
1. Descripción de implementación

El código desarrollado por el grupo cumple la función descrita en el enunciado, desde la invocación por terminal, que reconoce si el comando no está completo y guía al usuario para que ingrese bien los valores (NOTA: aunque el ejecutable se llama taller01, el código fuente y el ejecutable entregado si tienen el nombre correcto):



```
jclavijo@SISOPYAD: ~/Desktop/sistemasOperativos2330/taller1
jclavijo@SISOPYAD:~/Desktop/sistemasOperativos2330/taller1$ ./taller01 4 test4.txt 3
Seguro que ingresaste por consola los datos bien?
Recuerda que la estructura del comando debe ser:
./taller_procesos N1 archivo00.txt N2 archivo01.txt
DONDE:
N1 es el numero de elementos que tiene archivo00.txt NO SE TE OLVIDE EL .txt
archivo00.txt es uno de los 2 archivos que contiene datos
N2 es el numero de elementos que tiene archivo01.txt
archivo01.txt es el otro de los 2 archivos que contiene datos NO SE TE OLVIDE EL .txt
jclavijo@SISOPYAD:~/Desktop/sistemasOperativos2330/taller1$
```

Los componentes que hacen que este código funcione, principalmente son la recolección adecuada de los datos pasados desde la terminal,



```
63 int N1 = (int)atoi(argv[1]); //se almacenan los enteros N1 y N2 desde terminal
64 int N2 = (int)atoi(argv[2]);
65 char *archivo00 = argv[3]; //se almacenan los nombres de los archivos donde estan los enteros
66 char *archivo01 = argv[4];
67 int *vector0=(int*)malloc(N1*sizeof(int)); // se crean los vectores donde se almacenara la info de los archivos
68 int *vector1=(int*)malloc(N2*sizeof(int));
69 FILE *arch00; //se crean los datos de tipo archivo
70 FILE *arch01;
71 cargarPunteros(arch00,archivo00,vector0); //se llama a la funcion que pasa la informacion de los archivos a los vectores
72 cargarPunteros(arch01,archivo01,vector1);
73
```

Y la función que lleva a memoria los datos que los archivos tienen almacenados, llamada cargarPunteros() que recibe por parámetro el dato de tipo archivo, el nombre del archivo y el vector para almacenar los datos

```

16 //funcion que lee los archivos y carga a los vectores de punteros
17
18 void cargarPunteros(FILE *arch , char *archivo ,int *vect){
19     char contenido;
20     char *buffer = (char*)malloc(1024);
21     arch = fopen(archivo,"r"); //se abre el archivo
22     int i = 0;
23     //se lee el archivo
24     while (fgets(buffer, 1024, arch) != NULL) {
25         char *token = strtok(buffer, " ");
26         while (token != NULL) {
27             vect[i] = atoi(token);
28             i++;
29             token = strtok(NULL, " ");
30         }
31     }
32     fclose(arch);
33 }

```

Para cumplir la funcionalidad, dentro del main se hace todo el manejo de pipes y forks, como se puede ver en el código fuente, comentado.

Para hacer las pruebas correspondientes, se uso el creador de archivos trabajado en clase, para mantener unas pruebas realizables, solo se usaron 3 y 4 como los enteros que se debían almacenar en cada archivo, a continuación, se muestra la tabla de pruebas y los resultados de ejecución

Plan de pruebas: taller procesos			
Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido
1. Archivo test3.txt	968 218 675	1870	1870
2. Archivo test4.txt	393 51 649 777	1861	1861
3. Suma total	1870 1861	3731	3731

```

jclavijo@SISOPYAD: ~/Desktop/sistemasOperativos2330/taller1$ ./taller01 4 test4.txt 3 test3.txt
Suma del arreglo del fichero test4.txt: 1870
Suma del arreglo del fichero test3.txt: 1861
Suma de las 2 sumas enunciadas anteriormente: 3731

```

2. Conclusiones

En esta actividad de programación, hemos trabajado con un código que utiliza pipes y forks para lograr una comunicación eficiente entre procesos y realizar tareas concurrentes de manera exitosa. Al ver cómo nuestro código cumplió el plan de pruebas de manera satisfactoria, aprendimos la importancia de la planificación y la sincronización en entornos multi-hilo. Además, hemos adquirido una comprensión más profunda de cómo los procesos pueden interactuar a través de tuberías para compartir datos y realizar tareas complejas. Esta experiencia nos ha brindado valiosas lecciones sobre la programación en sistemas operativos y nos ha preparado para abordar desafíos similares en el futuro con mayor confianza y habilidad.