Proyecto - Lectura de archivos grandes

Manuel Catalán, Lester García, Edwin Hilario, Axel Rodríguez, Steven Villatoro

Universidad Rafael Landívar - Ingeniería en informática y sistemas

Resumen

Un sistema operativo es el programa esencial para cualquier equipo informático. Permite la interacción entre el usuario y el equipo en cuestión por medio de una interfaz gráfica fácil de entender y utilizar. Pero no solo se queda en la interacción con el usuario, también realiza tareas independientes las cuales brindarán una experiencia de uso más optimizada y ordenada de la que el usuario podría imaginar, como gestionar el software, controlar dispositivos externos, almacenar archivos, optimización de uso de memoria y disco, entre otros.

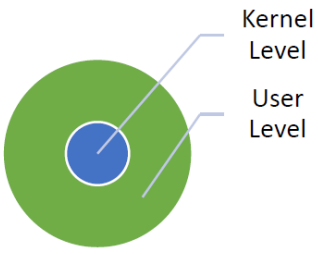
I. Introducción

En términos generales, un sistema operativo funciona como la capa intermedia entre la complejidad del hardware (parte física) y los requerimientos del software (parte interna) y los periféricos del equipo en cuestión.

Cuando se habla de un sistema operativo, no solo se queda en la parte gráfica con la cual interactúa el usuario, sino que también entran términos de ánalisis y control como lo son:

- Kernel
- Procesos
- Hilos (Threads)

entre otros.

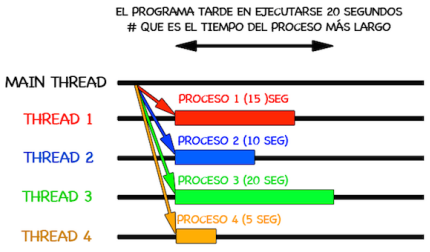


Cuando se habla de procesos y de hilos se está hablando de dos cosas totalmente diferentes, ya que un proceso es una entidad de ejecución independiente el cual al lanzarse proporciona espacios de memoria en los que dicho proceso podrá ejecutarse mientras que un hilo es una entidad la cual vive dentro de un proceso, son como "hijos"de un proceso más grande.

Aquí se puede observar mejor de forma gráfica:

Procesos

Hilos



II. Descripción del problema

La implementación de procesos distribuidos toma una importancia mayor cuando el

o los archivos que se buscan procesar son demasiado grandes y los recursos del equipo en cuestión para procesarlos son limitados.

III. Trabajo realizado

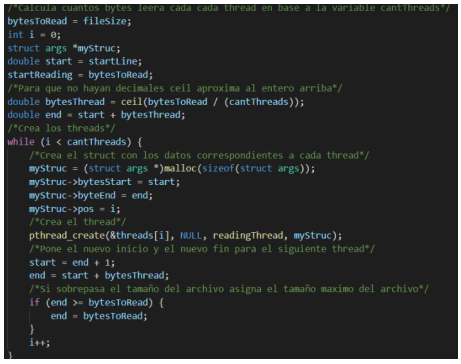
Para la realización del presente proyecto se hizo uso de una máquina virtual con una distribución de Linux instalada la cual fue Ubuntu en su versión 18.04.4. Al ser un sistema operativo de código abierto y libre se tiene la posibilidad de poder ingresar a opciones más avanzadas a las que un sistema Windows nos permitiría.

Se tuvo que investigar acerca del la manera de como leer archivos demasiado grandes, que son los threads dentro de un sistema operativo y para que como podríamos adaptarlos a nuestra solución y la eficiencia que utilizar threads conlleva dentro de una máquina con pocos recursos.

IV. Proceso y resultado final

Primero se lee el encabezado, se espera que el encabezado esté correcto por que al terminar de leer el encabezado se cuentan el número de columnas que tendrá el documento y después de leer este

Separamos los procesos de lectura del archivo en cuatro diferentes hilos. Los cuatro hilos son creados a principios antes de empezar a leer el documento. Se separa de esta manera: Cantidad de Bytes / Threads, entonces por ejemplo si el documento pesa 1 billón de bytes, se separan dentro 250 millones de bytes cada thread y mediante esto se hacen las lecturas por cada thread para que al finalizar de dividir cada uno se obtenga un archivo de salida tipo csv.




```
/*Espera a que los threads terminen*/
for (int j = 0; j < cantThreads; j++) {
    pthread_join(threads[j], NULL);
}
```

El proceso de cada thread es el mismo simplemente los parámetros de que reciben por cada uno serán diferentes por lo explicado anteriormente, ya que cada uno de ellos leerá en una posición diferente de bytes. Después de saber cuál es su posición inicial y su posición final, el thread empieza a recorrer el archivo. El thread le hace "Lock" la lectura del archivo para que ninguno de los otros threads estén leyendo al mismo tiempo el archivo. Que dos hilos diferentes esten leyendo el mismo archivo generaba un tipo error e investigando encontramos que a la hora de utilizar variables globales y no no les hacemos "Lock", bloquea el acceso a los otros threads estas variables globales hasta que termine un thread de utilizar el siguiente thread podrá utilizarlo. En esta parte se tiene un conteo interno array en donde todas las posiciones tiene un cero, se refiere al thread byte que se está leyendo. Cuando se encuentra cierto carácter, en conteo interno se dirige a su posición que es el tipo ASCII y suma las veces que encuentra cada carácter en ASCII.

```
/*Obtiene los argumentos del Struct*/
double bytePos = ((struct args*)input)->bytesStart;
double bytePosEnd = ((struct args*)input)->byteEnd;
int threadPos = ((struct args*)input)->pos;
char letras[400];
char c;
double conteoInterno[255];
int lastRead = 400;
/*Recorre el archivo dentro de los limites establecidos*/
for (double i = bytePos; i <= bytePosEnd; i += 400) {
    /*Calcula cuantos caracteres tiene que leer por iteración*/
    lastRead = ((i + 400) > bytePosEnd) ? (bytePosEnd - i) : 400;
    /*Bloquea el archivo para que solo el pueda acceder a el en el instante*/
    pthread_mutex_lock(&lock);
        /*Busca en el archivo, fseeko es para buscar en archivos grandes*/
        fseeko(archivoGrande, i, SEEK_SET);
        fread(&letras, 1, lastRead, archivoGrande);
    pthread_mutex_unlock(&lock);
    /*Recorre los caracteres leidos y suma en la posición correspondiente del array*/
    for (int i = 0; i <= lastRead; i++) {
        conteoInterno[(int) ((char) letras[i])]++;
    }
}
```

Se le hace un "Lock" a esta parte del código para que los otros threads tampoco los usen. Básicamente el conteo interno anterior que se tiene en esta parte se le iguala a la posición de la variable global conteo[i]. Al finalizar de leer todos los bytes del archivo, este se cierra. Por último se imprime en consola los valores encontrados en todo este proceso de los thread terminado y se recorre el arreglo de conteo y en su posición se convierte a carácter que es su código ASCII y muestra el número que contiene dicha posición del arreglo.

```
/*Asigna al contador global y verifica si todos los threads terminaron*/
pthread_mutex_lock(&lockVar);
    /*Suma al contador global*/
    for (int i = 65; i <= 90; i++) {
        conteo[i] += conteoInterno[i];
    }
    /*Asigna terminado a una posición correspondiente al thread*/
    terminados[threadPos] = true;
    bool finished = true;
    /*Verifica si todos los threads terminaron*/
    for (int i = 0; i < cantThreads; i++) {
        if (!terminados[i]) finished = false;
    }
    /*Si terminaron cierra el archivo, si no ignora eso*/
    if (finished) {
        fclose(archivoGrande);
    }
    /*Limpia la consola y muestra la cuenta de las letras*/
    //system("clear");
    for (int i = 65; i <= 90; i++) {
        printf("%c -> %lf\n", (char) i, conteo[i]);
    }
pthread_mutex_unlock(&lockVar);
```

Por último al terminar los threads en el método principal, se genera un archivo con el nombre output.csv que es donde se escriben los valores encontrados y se finaliza cerrando este archivo después de escribir los valores.

```
/*Obtiene la ubicación del proyecto actual*/
char cwd[1000];
getcwd(cwd, sizeof(cwd));
strcat(cwd, "/output.csv");
archivoGrande = fopen(cwd, "w");
char numStr[32];
char cadenaEscribir[100];
char letra[1];
for (int i = 65; i <= 90; i++) {
    letra[1] = (char) i;
    sprintf(numStr, "%f", conteo[i]);
    strcat(cadenaEscribir, letra);
    strcat(cadenaEscribir, ",");
    strcat(cadenaEscribir, numStr);
    strcat(cadenaEscribir, "\n");
    fwrite(archivoGrande, sizeof(cadenaEscribir), 1, archivoGrande);
}
fclose(archivoGrande);
```

V. Conclusiones

- Un sistema operativo trabaja como la capa intermediaria entre el usuario y todas las funciones que un equipo informático puede llegar a ofrecer
- La correcta utilización de procesos e hilos para la gestión de tareas dento de un sistema operativo puede influir mucho en el desempeño general del mismo.
- Pocos recursos no significa que sea imposible utilizarlo, solo hay que saber optimizarlo y se logrará obtener los mismos resultados e incluso mejores si se trabaja bien.

Referencias

[1] Sistemas operativos. https://www.areatecnologia.com/sistemas-operativos.htm

[2] Procesos. https://www.enciclopediadetareas.net/2012/06/caracteristicas de-los-procesos.html

[3] Procesos e hilos. https://www.webprogramacion.com/1/sistemas-operativos/procesos-e-hilos.aspx