SISTEMAS OPERATIVOS

PROCESAMIENTO CONCURRENTE DE ARCHIVOS

PROYECTO 2

Elaborado por: Ukranio Coronilla

El programa debe recibir dos parámetros, el primero es la ruta absoluta o relativa de una carpeta, la cual contiene m archivos de texto. El segundo es una letra que va a buscarse en los m archivos de texto. Al terminar el programa deberá imprimir para cada archivo dentro de la carpeta, cuantas coincidencias de la letra encontró.

Con intención de mejorar el rendimiento del programa, este deberá saber cuántos procesadores hay presentes en la computadora, y crear ese mismo número de procesos. Antes de crear los procesos, el proceso padre deberá haber implementado un ingenioso mecanismo, para repartir de la manera más equitativa posible los archivos a los procesos que se van a crear, y comunicárselos. Así cada procesador hará búsquedas solo en los archivos que le fueron asignados.

Sugerencias:

Recuerde que la información sobre el procesador se encuentra en el archivo `/proc/cpuinfo`. Para que el programa busque el numero en el archivo, puede utilizar las funciones de cadena `strstr()` y `strpbrk()`.

En UNIX para abrir un directorio y leerlo se utilizan las funciones `opendir()` y `readdir()`. A continuación se muestra un ejemplo de su uso, tomado de internet.

```
/* openreaddir.c by mind [mind metalshell com]
 *
 * Example on using opendir, closedir, and readdir to open a directory
 * stream and read in and print file names.
 *
 * 06/04/03
 *
 * http://www.metalshell.com/
 *
 */

#include <stdio.h>
#include <sys/types.h>
#include <dirent.h>
#include <errno.h>
```

```
int main(int argc, char *argv[])
{
        DIR             *dip;
        struct dirent   *dit;

        int             i = 0;

        /* check to see if user entered a directory name */
        if (argc < 2)
        {
                printf("Usage: %s <directory>\n", argv[0]);
                return 0;
        }

        /* DIR *opendir(const char *name);
         *
         * Open a directory stream to argv[1] and make sure
         * it's a readable and valid (directory) */
        if ((dip = opendir(argv[1])) == NULL)
        {
                perror("opendir");
                return 0;
        }

        printf("Directory stream is now open\n");

        /*  struct dirent *readdir(DIR *dir);
         *
         * Read in the files from argv[1] and print */
        while ((dit = readdir(dip)) != NULL)
        {
                i++;
                printf("\n%s", dit->d_name);
        }

        printf("\n\nreaddir() found a total of %i files\n", i);

        /* int closedir(DIR *dir);
         *
         * Close the stream to argv[1]. And check for errors. */
        if (closedir(dip) == -1)
        {
                perror("closedir");
                return 0;
        }

        printf("\nDirectory stream is now closed\n");
        return 1;
}
```

Notas finales

El proyecto es para entrega individual, cualquier duda acuda con el profesor, no lo deje al último y evite copiar código pues tanto el que copia como al que se le copió tendrá cero en la calificación.

Debe imprimirse el número de núcleos encontrados, el archivo, el PID del proceso que hizo la búsqueda, y el número de coincidencias.

Ejemplo de ejecución:

```
$ ./proyecto2 /home/usuario/Directorio s
Se encontraron 4 núcleos.

ARCHIVO               PID       NO. CARACTERES 's' ENCONTRADOS
Archivo1.txt          35486                 78
Archivo2.c            2256                  354
Archivo3.cpp          1335                  24
Archivo4.java         6876                  4
Archivo14.txt         35486                 69
Archivo22.c           2256                  0
Archivo31.cpp         1335                  54
Archivo47.java        35486                 7
Archivo11.txt         2256                  96
Archivo22.c           6876                  120
Archivo34.cpp         1335                  632
Archivo48.java        6876                  825
```

El nombre del archivo a subir debe ser el nombre del alumno separado con guion bajo, materia (SO), grupo, numero de proyecto y extensión c. El no cumplir con estos requisitos provocará la disminución de la calificación.

Es posible que el proyecto no cumpla todos los requisitos (lo cual provoca una calificación menor a 10), pero un proyecto que no compila o no hace lo que se pide recibe una calificación de cero.

Ejemplo de un nombre de archivo:

`Juan_Perez_Molinar_SO_2CM1_2.c`