Instituto Tecnológico de Costa Rica
Escuela de Computación
Curso: IC6600 - Principios de Sistemas Operativos

Proyecto 1

El objetivo de este proyecto es comparar el rendimiento de realizar una tarea colaborativa utilizando múltiples procesos. Para ello se deberá desarrollar una versión multiprocesos del programa llamado *grep* que permite buscar múltiples palabras sobre un archivo e imprimir el párrafo en dónde aparecen dichas palabras.

En este proyecto se deben realizar múltiples pruebas para determinar la cantidad óptima de procesos para ejecutar este tipo de tarea sobre un único archivo muy grande.

Uso del programa

El programa *grep* recibe una expresión regular y una lista de archivos:

```
grep 'pattern1|pattern2|pattern3|...' archivo
```

Por omisión el programa debe mostrar la línea del archivo en donde se encuentra la expresión regular.

Manejo de expresiones regulares

El lenguaje C por sí mismo no maneja expresiones regulares, sin embargo se puede usar la librería "regex.h" que cuenta con las funciones *regcomp* y *regexec* que permiten realizar esta tarea. La función *regcomp* se debe ejecutar inicialmente y recibe la expresión regular ingresada por comando, luego la función *regexec* se debe ejecutar sobre cada una de las líneas del archivo.

Manejo de múltiples procesos

El programa debe contar con múltiples procesos que se encarguen de inspeccionar el archivo. Debido a que las búsquedas se realizan sobre archivos de gran tamaño no sería eficiente crear un único proceso. Es por ello que es necesario crear un conjunto compartido de procesos (pool de procesos) que incluya solo una cantidad limitada de procesos, y en donde cada proceso trabajará sobre un buffer propio en donde se debe leer una porción del archivo. Como parte del proyecto se deben realizar diferentes pruebas (con diferentes archivos grandes) para determinar un tamaño "óptimo" del pool de procesos. Se utilizará el tiempo de ejecución para determinar dicho tamaño óptimo.

En este caso se utilizará un pool de procesos estático, es decir, el programa creará la totalidad de los procesos al inicio del programa y únicamente utilizará estos para realizar todo el trabajo. Tal como usted debe suponer, a los procesos

que terminen de buscar en su porción del archivo deberán leer otra parte del archivo para realizar la búsqueda hasta terminar con todo el archivo. Se debe tener especial cuidado de que dos o más procesos lean la misma porción del archivo, o bien, que traten de escribir al mismo tiempo en la salida.

Note que al leer una porción del archivo puede suceder que una línea no sea leída completamente. Por eso cada proceso debe revisar si alguna línea se cortó, y actualizar el puntero de lectura del archivo (**fseek**) hasta el final de la última línea que se leyó completa.

Sincronización entre procesos

Los procesos se deberán sincronizar entre ellos mediante mensajes (NO sockets). La sincronización incluye determinar la posición de la última lectura al archivo y escribir a la salida. Para ello puede utilizar al padre como un proceso coordinador al cual los otros procesos realizan estas peticiones.

Análisis de resultados

Se deberán realizar varias pruebas cambiando la cantidad de procesos en el pool. Una vez que se calcule el tiempo de ejecución final, se deben crear tablas y gráficas en donde se muestre cúal cantidad de procesos brinda el mejor rendimiento para realizar la búsqueda de la expresión.

Cree un archivo de bitácora (logfile) desde el proceso principal que escriba una entrada por cada porción del archivo que se lee, indicando el subproceso que lo copió, y el tiempo que se duró. Utilice prefereriblemente un formato CSV para poder realizar el análisis de resultados desde Excel o R.

Consideraciones generales

- Todo el desarrollo del proyecto debe realizarse en lenguaje C y sobre ambiente Unix.
- No es permitido el uso de otras librerías adicionales a "regex.h" para programar el proyecto. Tampoco es válida la copia de porciones significativas de código desde Internet.
- El tamaño del buffer de lectura para cada proceso será de 8K (8192 bytes). No es válido leer un tamaño de buffer más grande y luego dividirlo en buffers de 8K. La idea es que cuando algunos procesos están leyendo otro procesen los datos.
- Los procesos deben ir alternando la lectura al archivo. No es válido calcular de antemano las porciones que cada proceso va a leer.
- Se debe realizar el procesamiento sobre un único archivo muy grande (no varios archivos pequeños).
- Se deben utilizar procesos hijos NO hilos de ejecución.

- Se deberá generar una documentación formal, en formato pdf, en donde se describan las diferentes etapas del desarrollo del proyecto, las decisiones de diseño que se tomaron, los mecanismos de programación utilizados, y los resultados de las diferentes pruebas al programa. Dicha documentación deberá incluir al menos las siguientes secciones:
  - Introducción
  - Descripción del problema (este enunciado)
  - Definición de estructuras de datos
  - Descripción detallada y explicación de los componentes principales del programa
  - Mecanismo de creación y comunicación de procesos
  - Análisis de resultados
  - Conclusiones
- El proyecto puede realizarse en grupos de dos estudiantes.
- No se permite la copia de código entre grupos de estudiantes, o entregar código realizado por estudiantes los semestres anteriores, tampoco es permitido utilizar librerías adicionales (desarrolladas por terceros), o código generado por motores de inteligencia artificial.
- Puede utilizar el sitio del proyecto Gutenberg para descargar archivos de texto muy grandes. En particular puede descargar Don Quijote, Divina Comedia, Los hermanos Karamazov, Les Misérables, Guerra y Paz, y Anna Karenina.