Tarea Programada

Objetivos:

- Poner en práctica la comunicación de varios procesos por medio de buzones
- Poner en práctica la teoría sobre File System, por medio del diseño e implementación de un sistema simple

Instrucciones: debe crear un programa en C o C++ que utilice llamados al sistema en Linux para conectarse varias páginas web de forma simultánea. Su programa debe realizar un request a la página y basado en la respuesta recibida, debe contar la cantidad de etiquetas html que ha encontrado. Es decir, si encuentra la etiqueta ***<header>*** debe contar cuantas veces aparece en la respuesta recibida y se debe aplicar lo mismo para cada etiqueta contenida en la respuesta.

Las tareas de analisis de respuestas de servidores y otras, se deben separar por medio de procesos, de manera que cada proceso lleve a cabo una tarea distinta. Se debe tener un proceso por cada página a “analizar”. Tome en cuenta que si se determina que se va a revisar la misma URL dos veces, se necesitan dos procesos separados que lleven a cabo la tarea. Entonces, algunos de estos procesos se van a encargar de recolectar información que luego va a ser analizada por un proceso separado para finalmente se almacenada en un sistema de archivos por otro proceso.

Además, se van a necesitar segmentos de memoria compartida para que los procesos se puedan comunicar entre si. Los procesos que realizan el conteo de etiquetas deben pasar sus resultados a otro proceso que va a determinar el total de etiquetas global.

Por último, se debe diseñar e implementar un sistema de archivos que permita guardar todos los resultados obtenidos por medio de todos los procesos analizadores y los resultados globales. Note que el programa no debe cerrar en este punto, pues el usuario puede pedirle al sistema de archivos que le permita revisar alguno de los resultados obtenidos.

Parte I

Primero, debe realizar un diseño del programa que va a realizar. Tome en cuenta explicaciones y diagramas de forma que su idea quede clara.

Una vez entregado este diseño, se debe implementar el programa. Al no haber un file system aún, su programa debe mostrar todos los resultados obtenidos, ya sea imprimiendo en pantalla de forma que se entienda el resultado mostrado o bien generando archivos reales en un folder.

Parte II

Una vez lista la primera parte, se debe trabajar en un diseño de un sistema de archivos que permita “almacenar” los resultados. Tome en cuenta tamaños de bloque, jerarquía, permisos, y cualquier otro tema relevante visto en el curso. Este diseño debe ser aprobado por el docente o asistente (se indicará en clase en su momento).

Una vez tenga aprobación debe implementar el sistema de archivos, basado en el diseño aprobado. Si requiere realizar cambios, debe consultar con el docente o asistente del curso.

Notas:

- El programa DEBE correr en Linux, es decir, no se pueden utilizar llamados al sistema de Windows. Además, debe indicar si utiliza Ubuntu o Fedora.
- El documento debe estar en una sección de memoria compartida, para ello utilice mmap. Tome este enlace como guía: https://man7.org/linux/man-pages/man2/mmap.2.html
- Se deben utilizar msgrcv y msgsnd de Linux, los puede revisar en este enlace: https://linux.die.net/man/2/msgsnd
- Si necesita realizar sincronización, cree su propia clase Semáforo. Utilice este enlace como referencia: https://man7.org/linux/man-pages/man2/semget.2.html
- Va a necesitar un “KEY” para ese semáforo, defínalo utilizando su carné estudiantil
- Los procesos solamente se pueden crear con fork, no se pueden utilizar otras estructuras o librerías.
- Debe trabajar en clases separadas. No se permite que todo el código esté en una sola clase o main.

- El trabajo es en parejas o individual.
- Para crear un socket que se conecte a las páginas web, puede opcionalmente crearlos en Python y luego embeber esta funcionalidad en C/C++ (10 puntos extra)

Entregables:

Diseño I: 21 de abril

Código + documentación I: 18 de mayo

Diseño II: 1 de junio

Código + documentación II: 29 de junio