Proyecto final de SSOO

Gestor de planificación de procesos en Linux.

Nerea Alamar Porter

Descripción del proyecto

La finalidad de este proyecto es la de programar un gestor de planificación de procesos en Linux, mostrando una lista con procesos que se encuentran en ejecución, junto a las propiedades de dichos procesos.

Los procesos que se mostrarán serán eligidos por el usuario en una lista donde aparecen todos los procesos en ejecución, tal y como se muestra en la imagen, el usuario podrá ir eligiendo los procesos deseados.

Para mayor comodidad hay un botón de “esconder proceso”, el cual nos permite eliminar un proceso de nuestra lista de procesos para permitir gestionar otros procesos pudiendo volver a añadirlo después.

El programa también debe incorporar algunas de las funcionalidades vistas en la asignatura como modificar la prioridad de los procesos tanto estática como interna (nice), cambiar la política de planificación utilizada o el número de CPUs de las que el proceso hará uso, además de poder eliminar/detener el proceso elegido.

Manual de utilización

El primer paso para utilizar la aplicación de gestión de procesos es ir eligiendo procesos en la lista que aparece en la esquina superior izquierda e ir añadiéndolos, hasta tener los procesos deseados en la lista de la parte inferior (*Lista de procesos*) donde podremos observar las propiedades de cada proceso (su nombre, PID, política de planificación, prioridad estática, prioridad interna nice, CPUs afines, % de CPU utilizado, % de memoria utilizada, estado del proceso y tiempo).

Para poder realizar alguna acción hay que introducir en la entrada de texto el número del PID del proceso que deseamos gestionar.

Tanto para cambiar la política de planificación, la prioridad estática y la prioridad interna (nice), hay que seleccionar la opción deseada de la lista y cambiarla pulsando el botón correspondiente.

Para modificar la afinidad del proceso hay una serie de botones en función del número de CPUs del que dispongamos, y podemos activar o desactivar los botones para permitir que el proceso se pueda ejecutar o no en dichas CPUs.

Además tenemos dos botones; eliminar proceso y esconder proceso. El primero detiene y elimina el proceso directamente (además de eliminarlo de la lista), y el segundo simplemente lo elimina de la lista para permitir tener más espacio para gestionar otros procesos, si deseamos volver a añadirlo sólo tenemos que volver a seleccionarlo de la lista de arriba.

Detalles técnicos

Para la realización de este proyecto he utilizado la interfaz gtk 3.0. Dicha interfaz me ha permitido utilizar una serie de botones, labels y listas para poder hacer la aplicación más intuitiva, simple y fácil de utilizar.

Funciones:

lista()

- Para crear la lista donde escoger los procesos que gestionar he implementado una función que crea un fichero donde almacena el resultado de una instrucción en el terminal; he utilizado la instrucción “ps axu” donde aparece la información sobre los procesos y sus propiedades.

- He creado varios arrays globales donde he ido almacenando los datos para cada proceso de la lista.

add_proc()

- Una vez creada la lista, el botón de añadir proceso llama a una función que comparará la salida de texto de la lista con la lista global de PIDs (*lista_procesos*) y añadirá dicho proceso a una lista global de procesos de la aplicación, es decir, los procesos que aparecen en la lista inferior (*procesos*). Una vez hecho esto llamará a otra función *crear_caja()* que creará/imprimirá la línea con el proceso y su información.
- Cuando se llama a la función *eliminar_proceso()* o *esconder_proceso()* dicho proceso se eliminará de la lista de procesos global. Para evitar que queden huecos en la lista, en *add_proc()* se tiene en cuenta si en *procesos* hay algún hueco y añade el siguiente proceso en su lugar.

cambiar_afinidad()

- Función que se activa al activar o desactivar los botones de las CPUs, accede al proceso y mediante el código dado en la asignatura, aplica a dicho proceso las CPUs de los botones que estén activos.

cambiar_nice()

- Al pulsar el botón de cambiar la prioridad nice se accede a una función que lee la salida de la lista de nice y, si el proceso elegido tiene una política de planificación CFS se modifica, sino no hará nada, ya que ni RR ni FIFO tienen nice.

cambiar_prio_estatica()

- Al pulsar el botón de cambiar prioridad estática se accede a una función que lee la salida de la lista de prioridad estática y, si el proceso elegido tiene una política de planificación FIFO o RR se modifica su prioridad estática llamando a la función

cambiarPrioridadEstatica() que utiliza las instrucciones dadas en la asignatura para cambiar la prioridad.
- En caso de que el proceso tenga una política de planificación CFS no hará nada, ya que CFS tiene una prioridad estática de 0.

cambiar_planificador()
- Al pulsar el botón de cambiar la política de planificación se accede a una función que lee la salida de la lista de planificadores y cambiará la politica de planificación en función del valor seleccionado, además se calcularán las correspondientes prioridades estáticas y nice, dependiendo de si se trata de RR, FIFO o CFS.
- Para esto utilizará dos funciones, *getPlan()* que devolverá la política de planificación que está utilizando el proceso, y *cambiarPlan()* que se encargará de cambiar la política de planificación.

Las funciones para cambiar la prioridad estática, nice, afinidad y la política de planifiación son funciones muy similares a las hechas en ejercicios de prácticas del tema 4 de la asignatura por lo que no he tenido mucha dificultad para implementarlas.

Las funciones *crear_caja(), nombres_variables()* y el *main()* son prácticamente todo código de la interfaz, en el que tampoco voy a entrar mucho, ya que no está relacionado con la asignatura:
- *crear_caja()* → imprime en la aplicación las líneas con el proceso y sus propiedades.
- *nombres_variables()* → imprime los nombres de las variables del proceso y sus propiedades.
- *main()* → crea todo lo que es la interfaz, labels de texto, listas y botones.

Conclusión

Para empezar, considero que los objetivos marcados para el proyecto han sido alcanzados, después de varias versión y solucionar problemas, creo que la aplicación funciona correctamente y cumple con las expectativas que me había generado en un principio.

Para la realización del proyecto he necesitado recoger bastante información tanto de la asignatura como de internet para obtener la información que necesitaba de los procesos y también para aprender a utilizar la interfaz gráfica, ya que al empezar el proyecto no tenía mucha idea de cómo realizar una aplicación en C.

Para finalizar, desde un punto de vista positivo es la cantidad de información aprendida, tanto en la sección de búsqueda de información como en las funcionalidades aprendidas durante la asignatura (modificar las prioridades estáticas y nice, la afinidad y la política de planifiación), ya que gracias a este trabajo creo que me ha quedado todo mucho más claro. Además, el aspecto visual del programa está bastante bien logrado y desde mi punto de vista bastante organizado e intuitivo.

Bibliografía

http://www.tldp.org/
https://developer.gnome.org/
Temas 1 y 4 de VJ1225 en la Universidad Jaume I.