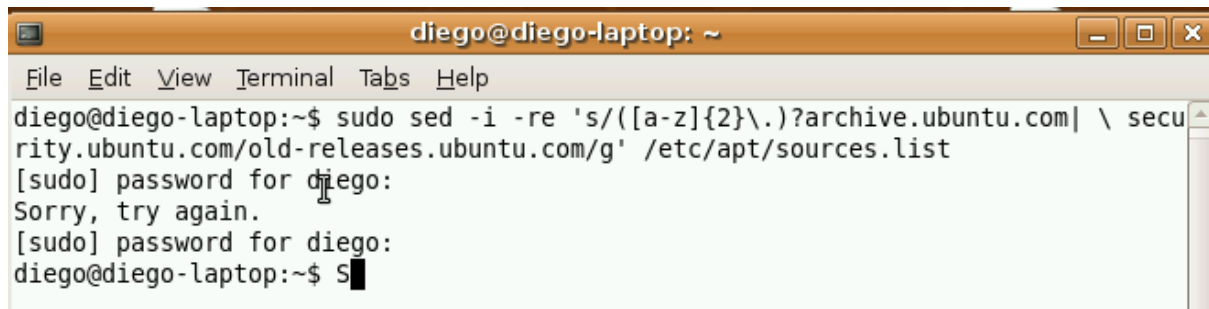


# Lab 5



```
diego@diego-laptop: ~  
File Edit View Terminal Tabs Help  
diego@diego-laptop:~$ sudo sed -i -re 's/([a-z]{2}\.)?archive.ubuntu.com| \ security.ubuntu.com/old-releases.ubuntu.com/g' /etc/apt/sources.list  
[sudo] password for diego:  
Sorry, try again.  
[sudo] password for diego:  
diego@diego-laptop:~$ S
```



```
Archivo Editar Ver Terminal Solapas Ayuda  
linux-2.6.24/sound/usb/usbaudio.c  
linux-2.6.24/sound/usb/usbaudio.h  
linux-2.6.24/sound/usb/usbmidi.c  
linux-2.6.24/sound/usb/usbmixer.c  
linux-2.6.24/sound/usb/usbmixer_maps.c  
linux-2.6.24/sound/usb/usbquirks.h  
linux-2.6.24/sound/usb/usx2y/  
linux-2.6.24/sound/usb/usx2y/Makefile  
linux-2.6.24/sound/usb/usx2y/usX2Yhwdep.c  
linux-2.6.24/sound/usb/usx2y/usX2Yhwdep.h  
linux-2.6.24/sound/usb/usx2y/usb428ctrldefs.h  
linux-2.6.24/sound/usb/usx2y/usbux2y.c  
linux-2.6.24/sound/usb/usx2y/usbux2y.h  
linux-2.6.24/sound/usb/usx2y/usbux2yaudio.c  
linux-2.6.24/sound/usb/usx2y/usx2y.h  
linux-2.6.24/sound/usb/usx2y/usx2yhwdeppcm.c  
linux-2.6.24/sound/usb/usx2y/usx2yhwdeppcm.h  
linux-2.6.24/usr/  
linux-2.6.24/usr/.gitignore  
linux-2.6.24/usr/Kconfig  
linux-2.6.24/usr/Makefile  
linux-2.6.24/usr/gen_init_cpio.c  
linux-2.6.24/usr/initramfs data.S  
diego@diego-laptop:/home/scheduler_dev$
```

## Resumen

- Funcionamiento y sintaxis de uso de structs.  
Son tipos de datos derivados, esto quiere decir que son construidas utilizando objetos de otros tipos. Al terminar de declararse debe llevar ; a su vez no pueden existir miembros con el mismo nombre. Pueden haber estructuras dentro de estructuras, pero no puede crear una instancia de sí misma.
- Propósito y directivas del preprocesador.  
El CPP es el procesador para C. Es el primer programa invocado por el compilador y procesa directivas como: #if, #elif, #else, #undef, #warning, #region, #error, #line, #endregion, #pragma

Éstas son las encargadas de ver la información del documento antes de completar la compilación verificando que no hayan errores.

- Diferencia entre \* y & en el manejo de referencias a memoria (punteros).  
\* se utiliza para obtener el contenido del puntero  
& se utiliza para obtener la dirección de memoria de la variable
- Propósito y modo de uso de APT y dpkg

APT: Es una librería que contiene la aplicación central del sistema operativo, en ella se pueden agregar o eliminar paquetes, se debe actualizar antes de hacer algún cambio.

Dpkg: Este es un manejador de paquetes de Debian de nivel medio. Es usado para instalar, borrar o gestionar paquetes de Debian/ Linux

### Preguntas

- ¿Cuál es el propósito de los archivos sched.h modificados?  
En ellos se definen estructuras las cuales contienen parámetros de programación necesarios para la implementación de cada política de programación soportada.
- ¿Cuál es el propósito de la definición incluida y las definiciones existentes en el archivo?  
SCHED\_NORMAL Es la programación estándar de los procesos en cualquier sistema Linux, emplea un sistema de tiempo compartido. Este asigna un tiempo para cada proceso.  
SCHED\_FIFO Esta política programa los procesos en torno al tiempo en el que llegan  
SCHED\_RR Esta política programa a los procesos dándoles una fija cantidad de tiempo para su ejecución  
SCHED\_BATCH Esta política es usada para ejecutar lotes de procesos, es similar al NORMAL. Este lidiar con procesos con los que no se pueden interactuar.  
SCHED\_IDLE El objetivo de esta política es encargarse de los procesos con baja prioridad. Estas se ejecutan cuando no hay nada más que ejecutarse.  
SCHED\_CASIO Su objetivo es administrar tareas en tiempo real programadas en función del algoritmo EDF.
- ¿Qué es una task en Linux?  
Toda información que viaja entre el CPU y el task\_entity es una tarea.
- ¿Cuál es el propósito de task\_struct y cuál es su análogo en Windows?  
Este contiene toda la información respectiva de un proceso que el kernel necesitara saber de este. En windows cada proceso es representado por EPROCESS (estructura) y KPROCESS (estructura) contiene la información para el kernel.
- ¿Qué información contiene sched\_param?  
Es una estructura que contiene la forma en la cual los procesos serán programados (tiempo).
- ¿Para qué sirve la función rt\_policy y para qué sirve la llamada unlikely en ella?  
rt\_policy eligira la política de programación (tiempo) que se llevará a cabo.  
Unlike es la sugerencia para el compilador, con ella el compilador usará las instrucciones para hacer funcionar el lado improbable del condicional.
- ¿Qué tipo de tareas calendariza la política EDF, en vista del método modificado?  
Con el uso de esta política sabremos cuando un evento de programación ha sufrido algún cambio. Se buscará entre el stack de procesos el más cercano a terminar, y se programara para ejecutarse. El tipo de tareas que se programan (tiempo) son las manejadas por casio.
- Describa la precedencia de prioridades para las políticas EDF, RT y CFS, de acuerdo con los cambios realizados hasta ahora  
Ahora Casio tiene la prioridad o se ejecuta antes que RT y CFS.
- Explique el contenido de la estructura casio\_task.

Contiene la estructura para cada nuevo nodo del árbol Red-Black, los nuevos tendrán una etiqueta, `absoulte_deadline`. Las otras dos estructuras son de los nodos ya recorridos y las tareas que manejan.

- Explique el propósito y contenido de la estructura `casio_rq`.  
Coordinar los procesos por prioridad, en función de ordenar la lista de procesos de forma ascendente. Esta será ordenada y revisada constantemente.
- ¿Qué es y para qué sirve el tipo `atomic_t`? Describa brevemente los conceptos de operaciones RMW (read-modify-write) y mapeo de dispositivos en memoria (MMIO).

**atomic\_t** es una variable de tipo entero que su propósito es implementar un tipo y un set de funciones que pueden operar en él de forma correcta por diferentes hilos sin requerir de sincronización.

**RMW** es un tipo de operador de `atomic_t`, este lee la ubicación en memoria y escriben nuevos valores de forma simultánea. A través de estos operadores se evitan las race-conditions.

**MMIO** es una redirección de memoria que permite construir el sistema de manera que el cpu utilice los registros del chip como si fueran posiciones en la memoria RAM.

- ¿Qué indica el campo `.next` de esta estructura?  
La dirección de memoria de la variable, la variable ingresada es la tarea ejecutable de un módulo, llegará en un orden decreciente.
- ¿Cuándo preempta una `casio_task` a la task actualmente en ejecución?  
En el momento de llamar a la función `Pick_Next_Casio` esta selecciona la tarea que se ejecutará en el proceso actual.
- ¿Qué información contiene el archivo `system` que se especifica como argumento en la ejecución de `casio_system`?  
Este contiene la duración de las tareas.
- Investigue el concepto de aislamiento temporal en relación a procesos. Explique cómo el calendarizador `SCHED_DEADLINE`, introducido en la versión 3.14 del kernel de Linux, añade al algoritmo EDF para lograr aislamiento temporal.

El aislamiento de procesos es un mecanismo para ejecutar programas con seguridad y de forma separada. Es utilizado para ejecutar nuevo código de una fuente no confiable, a través del aislamiento se puede controlar los recursos proporcionados a los programas.

`SCHED_DEADLINE` está basado en el algoritmo Edit Deadline First, este reserva recursos. Las tareas son creadas independientemente de sus requisitos de tiempo, el kernel los acepta luego de realizar una prueba de calendarización. El aislamiento temporal entre tareas se da cuando una tarea intenta ejecutarse por más tiempo del asignado, el algoritmo suspenderá la tarea y aplaza su ejecución hasta cuando se vuelva a activar.