* Funcionamiento y sintaxis de uso de structs.

Las matrices permiten definir el tipo de variables que pueden contener varios elementos de datos del mismo tipo. Del mismo modo, la **estructura** es otro tipo de datos definido por el usuario disponible en C que permite combinar elementos de datos de diferentes tipos.

El formato de la declaración de estructura es el siguiente:

struct [structure tag] {

member definition;

member definition;

...

member definition;

} [one or more structure variables];

* Propósito y directivas del preprocesador

El propósito de las directivas es el manejo de la información de algún archivo para así, antes de compilarlo, vea que no haya errores. El preprocesador de C es lo primero en ser invocado por el compilador y así procesar las directivas como los #includes.

* Diferencia entre \*y &en el manejo de referencias a memoria (punteros).

El & sirve para poder obtener la referencia de un espacio de memoria a traves de la dirección de memoria. Es como obtener la dirección de memoria de una variable. Además, que el \* es para declarar punteros o hacer referencia a espacios de memoria reservados para alguna variable en específico.

* Propósito y modo de uso de APT y dpkg.

APT es un manejador de paquetes el cual se encarga de la instalación o eliminación de archivos y paquetes dentro de Linux. Con esta herramienta se hace mas simple el instalar, eliminar o gestionar algún paquete, además de la instalación de dependencias, lo cual se encarga APT. Por otro lado, tenemos a dpkg que solamente se encarga de instalar paquetes .deb que se le indican usando el comando. Este a diferencia de APT muestra las dependencias necesarias para poder llevar a acabo la instalación, pero este no las instalara y tampoco configurara nada al no haber dichas dependencias.

* ¿Cuál es el propósito de los archivos sched?h modificados?

Es un archivo el cual modificamos para así poder implementar las políticas de calendarización al compilar un programa que queremos usar.

* ¿Cuál es el propósito de la definición incluida y las definiciones existentes en el archivo?

El propósito de la definición incluida el crear una nueva política que sea aceptable para realizar la configuración de la calendarización. En las definiciones existentes del archivo tienen el propósito de permitir utilizar la política definida para hacer uso de una serie de directivas para aplicar la calendarización que hicimos.

* ¿Qué es una task en Linux?

Los task se pueden ver como una unidad de ejecución que pueden compartir sus recursos del sistema con otros task. Es como ver un thread.

* ¿Cuál es el propósito de task\_struct y cuál es su análogo en Windows?

El task\_struct, por lo que creo que he entendido, task\_struct es la estructura C que actúa como el descriptor del proceso, que contiene todo lo que el núcleo podría necesitar saber sobre los procesos. Al final del proceso, la pila del núcleo vive otra estructura, thread\_info, que tiene un puntero a los procesos task\_struct. En el caso análogo de Windows seria Win32PrioritySeparation.

* ¿Qué información contiene sched\_param?

Es contiene la información de un proceso o thread asignado al calendarizado. Este posee parámetros dentro de el y con él, el proceso que esta ejecutando y la priorización.

* ¿Para qué sirve la función rt\_policy y para qué sirve la llamada unlikely en ella?

Esta función sirve para hacer una verificación y ver si el tipo de política de calendarización que se quiere hacer es valida dentro de la configuración de la calendarización en tiempo real. Luego la llamada unlikely ayuda a indicar al compilador la rama del programa que quiere sea más rápida y el cuál puede ser ejecutada mas lentamente, lo cual hace el unlikely en este caso.

* ¿Qué tipo de tareas calendariza la políticaEDF, en vista del método modificado?

En lo que hemos modificado, pues tratamos de que calendarice las tares ejecutadas en tiempo real. Así el algoritmo EDf prioriza de manera mas dinámica. Así se colocan los procesos a una cola que prioriza el proceso mas cercano a su tiempo limite en tiempo real.

* Describa la precedencia de prioridades para las políticas EDF, RT y CFS, de acuerdo con los cambios realizados hasta ahora.
  + El EDF es el que este encargado de la prioridad principal de uso de las políticas y las ejecuta respecto a límite de cada proceso.
  + El RT este pasa de estar de las políticas del rt\_sched\_class a estar con los campos dados a las políticas del EDF para ver la prioridad de cada proceso.
  + El CFS este funciona como el planificador predeterminado en caso falle alguna política. Esta meja la asignación de recursos para el mayor rendimiento del procesador.
* Explique el contenido de la estructura casio\_task.

Esta primero crea una variable estructura del tipo red-black de tipo nodo. A esta se crea una variable deadline de tipo long long. De ahí se crea una variable del tipo struct para guardar nodos y al final, una nueva variable estructura que contendrá los parámetros de la tarea asignada al calendizador.

* Explique el propósito de la estructura casio\_rq.

Esta es crea una estructura que permita tener cierta variable del tipo red-black tree, además de tener una lista head de calendarización casio y finalmente llevar la variable nr\_running. Así guarda la posición y el estado del task.

* ¿Qué indica el campo .next de esta estructura?

Este campo indica el tipo de calendarización que deber ir después de la prioridad actual.

* ¿Cuándo preemptea una casio\_task a la task actualmente en ejecución?

Cuando hay un proceso de tipo casio en cola de ejecución y se quiere asignar una tarea que no es de su tipo, es que ocurre se preemptea.

* ¿Qué información contiene el archivo system que se especifica como argumento en la  
  ejecución de casio\_system?

Dentro posee la configuración de los task que se pararan a la calendarización del casio\_system, que contiene el tiempo de duración y la información del task.

* Investigue el concepto de aislamiento temporal en relación a procesos.

Este concepto nos habla de un mecanismo hecho para poder ejecutar programas con seguridad de manera separada a los de sistema principal. Esto bien al utilizar o ejecutar código nuevo o por aplicaciones de poca confiabilidad ajenos del sistema operativo. Así se tiene una forma de lograr que no interfiera con respecto a las limitaciones temporales entre ellos.

* Explique cómo el calendarizador SCHED\_DEADLINE, introducido en la versión 3.14 del  
  kernel de Linux, añade al algoritmo EDF para lograr aislamiento temporal.

Este calendarizador del procesador que se encuentra en el kernel de Linux, se basa en el algoritmo de EDF. Ahora este nueva versión a implementado en el SCHED\_DEADLINE el algoritmo de CBS que realiza el aislamiento de tasks temporal.