Universidad del Valle de Guatemala Andrea María Cordón Mayén

Sistemas Operativos **Carné:** 16076

Tomás Gálvez Peña **Fecha:** 25/02/2019

**Laboratorio #4**

**Preguntas:**

* **Funcionamiento y sintaxis de uso de structs**
  + La palabra reservada struct indica que se está definiendo una estructura. Una estructura es una colección de variables relacionadas bajo un nombre en específico. Las estructuras pueden tener variables de muchos tipos diferentes de datos. Para definir un struct, se debe seguir la siguiente sintaxis:

struct estructura {

char i;

int j;

…

}

* + El funcionamiento principal de struct es definir “estructuras” o variables que contienen distintos tipos de datos. Es decir que un struct puede tener un entero, un caracter o un string sin ningún problema.
* **Propósito y directivas del preprocesador:**
  + El preprocesamiento es la primera etapa en la compilación de programas en C. Entonces, un preprocesador es el que se encarga preprocesar el código fuente para que el compilador lo pueda procesar y, como su nombre lo indica, compilar. En cuanto a las directivas del preprocesador, tienen varias ventajas. Por ejemplo, estas pueden facilitar el desarrollo de software, facilitar la lectura de programas y facilita ocultar información dentro de un programa. Ejemplos de estas directivas son: #define, #elif, #else, etc.
* **Diferencia entre \* y & en el manejo de las referencias a memoria (punteros)**
  + \* permite declarar un tipo de dato puntero, pero también permite ver el valor que se encuentra en la dirección asignada. Por otro lado, & permite simplemente acceder a la dirección de memoria de una variable, pero no a su valor.
* **Propósito y modelo de uso de APT y dpkg**
  + APT es un manejador de paquetes del sistema que ayuda a manejar la instalación de paquetes en Linux. Por otro lado, dpkg no solo instala un paquete, sino que también va a notificar de cualquier dependencia que necesite ser instalada, pero no las va a instalar y no las va a configurar.
* **¿Cuál es el propósito de los archivos sched.h modificados?**
  + Configurar nuestras propias task o dejar preparado el archivo para que las pueda recibir y compilar.
* **¿Cuál es el propósito de la definición incluida y las definiciones existentes en el archivo?**
  + Contener el valor de las otras calendarizaciones y agregar una prioridad nueva.
* **¿Qué es un task en Linux?**
  + Todo lo que va a a ser procesado en algún momento por el CPU es una task. Por ejemplo, un proceso es una task, un thread también es una task.
* **¿Cuál es el propósito de task\_struct y cuál es su análogo en Windows?**
  + Task struct tiene como propósito declarar cuál es la estructura que va a tener una task. Es decir, qué datos van a conformar una task/proceso. Su análogo en Windows sería EPROCESS que es como una mezcla de task\_struct y mm\_struct
* **¿Qué información contiene sched\_param?**
  + Contiene bastante información acerca de las task programadas, pero una de las más importantes es la prioridad de ejecución que tiene cada task.
* **¿Para qué sirve la función rt\_policy y para qué sirve la llamada unlikely en ella?**
  + Normalmente es utilizada para decidir si una política de calendarización (scheduling policy) pertenece a la real-time class o no. Unlikely verifica si la política de una calendarización es alguna en específico o si pertenece a alguna de estas políticas.
* **¿Qué tipo de tareas calendariza la política EDF, en vista del método modificado?**
  + Calendariza todas las tareas que tienen una fecha de entrega (deadline) definida.
* **Describa la precedencia de prioridades para las políticas EDF, RT y CFS de acuerdo con los cambios realizados hasta ahora**
  + EDF es la política de deadline,, RT es la política que indica si la task debe correr en tiempo real y CFS
* **Explique el contenido de la estructura casio\_task**
  + Casio\_task tiene varios structs dentro de su estructura y en ellos se definen varios aspectos de las task que se están programando o se van a programar. Por ejemplo, hay un struct para la fecha de “entrega o realización”, uno para la task en sí, etc. Es importante mencionar que este tipo de task siempre se mantienen presentes en el sistema.
* **Explique el propósito de la estructura casio\_rq** 
  + Según comprendo, el propósito es hacer una llamada a las task que hayan sido programadas.
* **¿Qué indica el campo .next de esta estructura?**
  + Hace una llamada a la siguiente task que se encuentre programada.
* **Tomando en cuenta las funciones para manejo de lista y red-black tree de casio\_tasks, explique el ciclo de vida de una casio\_task desde el momento en el que se le asigna esta clase de calendarización mediante sched\_setscheduler. El objetivo es que indique el orden y los escenarios en los que se ejecutan estas funciones, así como las estructuras de datos por las que pasa. ¿Por qué se guardan las casio\_tasks en un red-black tree y en una lista encadenada?**
* **¿Cuándo preemtea una casio\_task a la task actualmente en ejecución?**
  + Cuando tiene una prioridad mayor a la task que está actualmente en ejecución.