Universidad del Valle de Guatemala
Sistemas Operativos - Sección 10

Christopher Sandoval 13660
24/02/2020

Laboratorio 4

- Funcionamiento y sintaxis de uso de structs.
  Un struct es similar a una estructura de una clase donde se pueden almacenar atributos en un solo conjunto para referenciarlos más adelante.

  La sintaxis es como la siguiente:

```
struct struct_name {
   DataType member1_name;
   DataType member2_name;
   DataType member3_name;
   ...
};
```

- Propósito y directivas del preprocesador.
  Estas son instrucciones que se ejecutan al compilar el programa. Existen varias directivas como por ejemplo #if que le indican al compilador si incluir ese segmento de código si la condición se cumple. Dependiendo del compilador que directivas podremos usar.

- Diferencia entre * y & en el manejo de referencias a memoria (punteros).
  El * se utiliza para declarar un puntero. & se utiliza para referenciar al valor de una variable.

- Propósito y modo de uso de APT y dpkg.
  El APT es un manejador de paquetes de linux. Con el podemos instalar, actualizar y eliminar paquetes. Su uso más común es apt-get install para poder descargar e instalar un paquete nuevo. La diferencia entre APT y dpkg es que con dpkg se manejan otro tipo de paquetes llamados .deb. Tiene instrucciones similares al apt y estas se manejan con directivas como -i para instalar.

- ¿Cuál es el propósito de los archivos sched.h modificados?
  Este nos sirve para definir los el nuevo tipo de scheduler que utilizaremos. En el hacemos referencia a las funciones que utilizara el nuevo scheduler.

- ¿Cuál es el propósito de la definición incluida y las definiciones existentes en el archivo?
  Incluimos una nueva definición que le indicará al sistema operativo el nuevo calendario a utilizar. Las definiciones que ya se encuentran en este archivo son pertenecientes al sistema de calendarización que tiene actualmente el sistema.

- ¿Qué es una task en Linux?
  Una task es el nombre que se le da a una unidad de ejecución en el sistema operativo Linux.

- ¿Cuál es el propósito de task_struct y cuál es su análogo en Windows?
  El task_struct es el que contiene toda la información de un proceso en Linux. En windows el análogo sería el task scheduler.

- ¿Qué información contiene sched_param?
  El sched_param contiene todos los parámetros necesarios para que funcione la política de calendarización.

- ¿Para qué sirve la función rt_policy y para qué sirve la llamada unlikely en ella?
  Esa función sirve para establecer qué política de de calendarización se utilizara en el sistema operativo y como se determinara el orden de los procesos. La llamada unlikely se utiliza para establecer que la condición establecida no se cumplira la mayoría de las veces, con propósitos de optimización.

- ¿Qué tipo de tareas calendariza la política EDF, en vista del método modificado?
  Esta política de calendarización se utiliza en una calendarización real time donde se prioriza la velocidad del cambio de tasks en el procesador.

- Describa la precedencia de prioridades para las políticas EDF, RT y CFS, de acuerdo con los cambios realizados hasta ahora.
  La política EDF es una política que se enfoca en tareas de alta prioridad con una política de tiempo real.
  La política RT maneja las tareas de baja prioridad con una política de tiempo real.
  La política CFS no es en tiempo real y maneja tareas de poca prioridad.

- Explique el contenido de la estructura casio_task.
  El casio_task es un proceso por lo que se le asigna una posición en el red-black tree, luego también se le debe asignar un deadline para que se pueda calcular su prioridad. Como todas las demás tasks se les debe indicar los mismos elementos de la estructura de cualquier proceso.

- Explique el propósito de la estructura casio_rq.
  El propósito de esa estructura es establecer el comportamiento del queue de procesos. Lo principal de la estructura se compone de un puntero hacia el árbol y una variable que indica que elemento se está ejecutando actualmente.

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

El campo .next es un puntero que se utiliza para ordenar los módulos de calendarización según las prioridades que manejan. En este caso se asignará casio_sched como el de mayor prioridad y el next será el rt_sched.

- 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. 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?
  Después de que se crea una casio_task se colocan los parámetros que se encuentran en sched_param y se inserta en el arbol asi como la lista encadenada. Cuando sea su hora de salir se elimina de la lista encadenada y se saca del árbol. Se guardan en este árbol para que se ejecuten en el orden deseado cuando ya sea momento de meter otra tarea al procesador.

- ¿Cuándo preemptea una casio_task a la task actualmente en ejecución?
  Cuando hay una task con un deadline más corto que el que se está ejecutando.

- Diferencia entre pre_casio.txt y new_casio.txt
  La diferencia es que en el pre_casio las tareas se ejecutan en un diferente orden que en el new_casio ya que el calendarizador es diferente.

- ¿Qué información contiene el archivo system que se especifica como argumento en la ejecución de casio_system?
  Contiene información de cómo funciona el sistema, estableciendo que calendarizadores tiene, prioridades, etc. Esta información queda expuesta al casio_system.

- Explique cómo el calendarizador SCHED_DEADLINE añade al algoritmo EDF para lograr aislamiento temporal.
  Se añade al SCHED_DEADLINE al algoritmo EDF para agregar el funcionamiento en el que si un task vence su tiempo de deadline entonces se le dará prioridad sobre la calendarización de otros procesos. En este caso se podría posponer la ejecución de un task hasta que le toque su turno nuevamente.