Universidad del Valle de Guatemala
Sistemas Operativos - Sección 10
María Fernanda Estrada 14198
24/02/2020

Laboratorio 4

- Funcionamiento y sintaxis de uso de structs.
  A diferencia de los arrays, los structs sirven para almacenar elementos de diferente tipo; son usualmente usados para representar un récord. Su sintaxis es como el siguiente ejemplo:

```
struct [structure tag] {

   member definition;
   member definition;
   ...
   member definition;
} [one or more structure variables];
```

```
struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
} book;
```

- Propósito y directivas del preprocesador.
  El preprocesador transforma el programa antes de la compilación del código. Es el primer programa invocado por el compilador y procesa directivas como #include, #define e #if, las cuales no son específicas de C. Las directivas son definidas para que el compilador realice algo antes de compilar el programa.

- Diferencia entre * y & en el manejo de referencias a memoria (punteros).
  * declara un puntero. & declara una referencia al dato.

- Propósito y modo de uso de APT y dpkg.
  APT (Advanced Packaging Tool) sirve para gestionar paquetes en sistemas Linux; permite instalar, actualizar o eliminar estos paquetes. Dependiendo de la acción, se pueden usar los siguientes comandos: apt-get, apt-cache, apt update, apt upgrade, apt install, apt remove, etc.
  Similar al APT, dpkg es utilizado para el manejo de paquetes, pero con la diferencia que puede gestionar archivos *.deb*. Algunos ejemplos de comandos son: dpkg -i, dpkg -r, dpkg -c, etc.

- ¿Cuál es el propósito de los archivos sched.h modificados?
  Estos archivos contienen los parámetros necesarios para la implementación de la política de calendarización que soporta. Al modificarla, agregamos un nuevo tipo de calendarizador.

- ¿Cuál es el propósito de la definición incluida y las definiciones existentes en el archivo?
  La definición incluida indica que agregamos un nuevo calendarizador llamado SCHED_CASIO. Las definiciones existentes indican qué calendarizadores está usando actualmente el sistema operativo, como FIFO, Round Robin, Batch, etc.

- ¿Qué es una task en Linux?
  El término hace referencia a una unidad de ejecución o de trabajo, las cuales pueden compartir recursos con otras tasks.

- ¿Cuál es el propósito de task_struct y cuál es su análogo en Windows?
  El propósito del task_struct es contener toda la información de un proceso. El análogo en Windows es el *Task Scheduler*.

- ¿Qué información contiene sched_param?
  Contiene los parámetros de calendarización requeridos para implementar la política de calendarización soportada.

- ¿Para qué sirve la función rt_policy y para qué sirve la llamada unlikely en ella?
  Sirve para decidir si una política de calendarización pertenece a la clase real-time (SCHED_RR y SCHED_FIFO) o no. La llamada unlikely es una sugerencia para el compilador que le permite optimizar al saber cuál es la menos probable.

- ¿Qué tipo de tareas calendariza la política EDF, en vista del método modificado?
  La política EDF calendariza tareas real-time.

- Describa la precedencia de prioridades para las políticas EDF, RT y CFS, de acuerdo con los cambios realizados hasta ahora.
  1. La política EDF maneja tareas de mayor prioridad, ya que son tareas de real-time.
  2. La política RT es de tiempo real, pero maneja tareas de menor prioridad.
  3. CFS no es real-time, por lo que sus tareas no tienen tanta prioridad como las dos anteriores.

- Explique el contenido de la estructura casio_task.
  Como es un proceso, se le deben definir los parámetros o la estructura que usará el proceso. Primero, al ser un proceso se le asigna un nodo en un árbol red-black. Luego, como la política es EDF, necesita un valor de deadline para asignarle prioridad. Después, usando list_head, se define un puntero al primer elemento de la lista casio. Por último, se da un puntero para indicar la estructura de cualquier proceso.

- Explique el propósito de la estructura casio_rq.
  El propósito es definir cómo se comportará el running queue de procesos. Se da un puntero hacia el inicio del árbol, otro puntero a la cabeza de una lista y por último un elemento que indica qué se está ejecutando.

- ¿Qué indica el campo .next de esta estructura?
  El campo .next es un puntero a sched_class que se usa para organizar los módulos de calendarización según la prioridad, comenzando por el de mayor prioridad. El de mayor prioridad en este caso es casio_sched_class y .next nos indica que el segundo de mayor prioridad es rt_sched_class.

- 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?
  Cuando se crea una casio_task, se le asignan los parámetros establecidos en sched_param. Si queremos que esté en fila para ejecución, se le asigna un deadline y se inserta en un red-black tree. Si ya tiene que salir de la lista de ejecución, se elimina del red-black tree y también de la lista encadenada.
  Las casio_tasks se guardan en un red-black tree porque debe mantener el balance del resto de tareas y se deben poder organizar automáticamente garantizando este balance. Se guardan en una lista encadenada porque deben tener un puntero referenciando a su información.

- ¿Cuándo preemptea una casio_task a la task actualmente en ejecución?
  Cuando existe una task con deadline menor a la que se está ejecutando actualmente.

- Diferencia entre pre_casio.txt y new_casio.txt
  Se diferencian en el orden que se ejecutan las tareas, basándose principalmente en el deadline en new_casio.txt

- ¿Qué información contiene el archivo system que se especifica como argumento en la ejecución de casio_system?
  Tiene la información del sistema, como calendarizadores, estructura de procesos, entre otras cosas. Toda esta información entonces la puede acceder casio_system.

- Explique cómo el calendarizador SCHED_DEADLINE añade al algoritmo EDF para lograr aislamiento temporal.
  En SCHED_DEADLINE, las tasks declaran independientemente sus requisitos de tiempo y el kernel las acepta en el calendarizador después de una prueba separada de calendarización. Si una task intenta ejecutarse más tiempo de lo requerido inicialmente, el kernel suspenderá esa task y pospondrá su ejecución hasta el siguiente periodo de activación.