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.
  x

- Explique el contenido de la estructura casio_task.
- Explique el propósito de la estructura casio_rq.
- ¿Qué indica el campo .next de esta estructura?
- 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?
- ¿Cuándo preemptea una casio_task a la task actualmente en ejecución?
- Diferencia entre pre_casio.txt y new_casio.txt
-