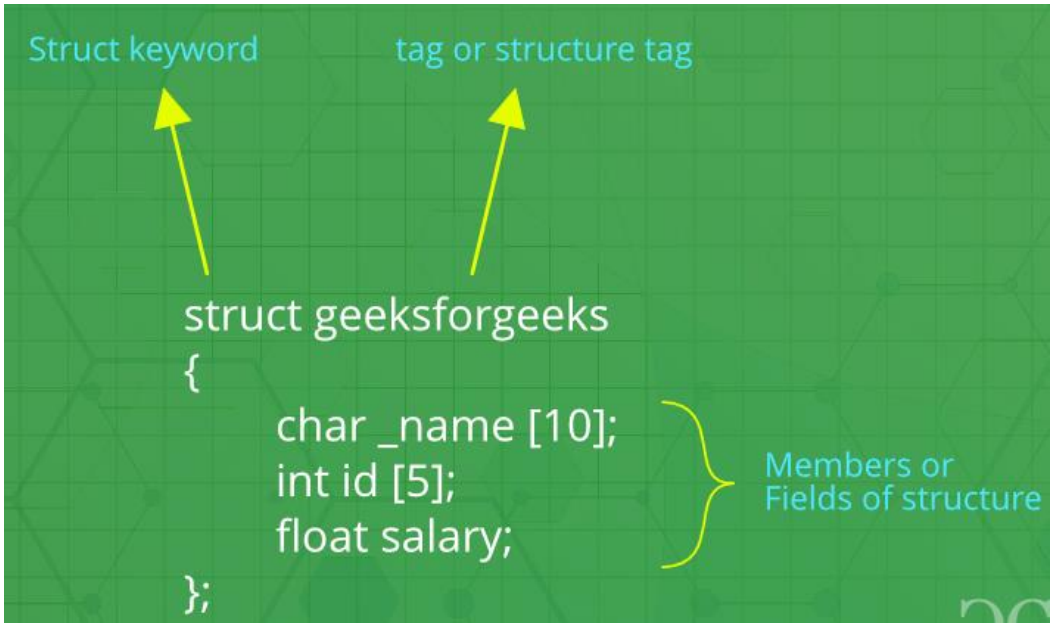
Universidad del Valle de Guatemala
Yasmin Chávez
Sistemas Operativos
Carné: 16101

Laboratorio #4

Investigue y resuma:

- **Funcionamiento y sintaxis de uso de structs.**
  Un struct es útil, porque permite definir un nuevo tipo de dato y puede estar compuesta por varios elementos e incluso de distintos tipos de datos. Para accesar a sus elementos se utiliza un punto (.) si tenemos un puntero hacia nuestro struct entonces se utiliza una flecha (->) para accesar a los elementos.
  Sintaxis:



- **Propósito y directivas del preprocesador.**
  El preprocesador es una herramienta que verifica el código antes que el compilador. Tiene 4 fases: tokenizado léxico, reemplaza la secuencia de trigrafos por los caracteres que representan; Empalmado de líneas, se forman líneas lógicas de forma se eliminan los saltos de línea en el código; Tokenización, reemplaza los comentarios por espacios en blanco. Divide cada uno de los elementos a preprocesar por un carácter de separación también realiza la expansión de macros y gestión de directivas.

  Las directivas son líneas de control que se comunican con el preprocesador. Deben iniciar con el símbolo #. Algunas de las directivas más comunes son:

  #define: Creación de constantes simbólicas y macros funcionales.

#undef: Eliminación de constantes simbólicas.
#include: Inclusión de ficheros.
#if (#else, #endif): Inclusión condicional de código.

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

  Con & obtenemos la dirección de memoria de una variable, nos devuelve un dato en hexadecimal. Los * definen a una variable como puntero, en estas variables se pueden almacenar direcciones de memoria. Por ejemplo:

  Int x = 7
  Int *pointer = &x

  En la segunda línea declaramos un puntero de tipo de dato int que almacena el la dirección de memoria de la variable x.

- **Propósito y modo de uso de APT y dpkg.**

  Apt es un sistema manejador de paquetes, es decir, es un conjunto de herramientas que permiten instalar, borrar, y cambiar paquetes. Su modo de uso es:

  apt acción package

  Donde acción puede ser install, remove, update, entre otros y package es el paquete al que se desea aplicar la acción.

  Dpkg es una herramienta para instalar, construir, borrar y manejar paquetes de la distribución Debian. También permite conocer información sobre los paquetes disponibles. La información esta dividida en tres clases: estados, selección de estados y banderas. Su modo de uso es similar al de apt:

  dpkg acción package

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

En estos archivos aparece información sobre cada uno de los procesos, fueron modificados para definir un macro para identificar “the scheduling policy”.

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

Los #define son las herramientas utilizadas para crear los macros y su propósito es realizar una sustitución de texto. Su estructura es:

#define identificador texto de sustitución

¿Qué es una task en Linux?

Es una tarea que debe ser procesada o un proceso.

¿Cuál es el propósito de task_struct y cuál es su análogo en Windows?

Task_struct es un descriptor de procesos que contiene toda la información sobre un proceso en específico. Su análogo en Windows es Task Scheduler.

¿Qué información contiene sched_param?

Es la estructura que describe los parámetros del scheduling. Contiene:

- sched_priority: la prioridad asignada a cada proceso o thread.
- Sched_curpriority: cuando se obtienen los parámetros del scheduling este valor cambia por la prioridad del thread o proceso que esta corriendo actualmente. Es el valor que utiliza el kernel cuando realiza decisiones en el scheduling.
- Sched_ss_low_priority: el thread de menor prioridad que se esta ejecutando.
- Sched_ss_max_repl: el máximo numero de veces que una tarea puede ser reemplazada, usualmente por bloqueos. Si se bloquea varias veces cambia a la prioridad más baja hasta que su presupuesto de ejecución sea reemplazado.
- Sched_ss_repl_period: el tiempo que se utiliza para reemplazar un presupuesto de una ejecución después de ser bloqueado o pasarse de su máximo numero de reemplazos.
- Sched_ss_init_budget: El tiempo que se utiliza como presupuesto de los threads.

¿Para qué sirve la función rt_policy y para qué sirve la llamada unlikely en ella?

Se utiliza para decidir si una scheduling policy pertenece a una clase de tiempo real (SCHED_RR and SCHED_FIFO) o no. La llamada unlikely es un macro que nos sirve como condicional para optimizar el compilador. En este caso verifica que policy sea igual a SCHED_CASIO y se utiliza unlikely por la probabilidad que este escenario ocurra.

¿Qué tipo de tareas calendariza la política EDF, en vista del método modificado?

Es un algoritmo de calendarización dinámico que asigna la prioridad más grande a la tarea con la caducidad más temprana. Uno de los elementos importantes es la caducidad y se cambia en la estructura de datos de sched_param. Casio_id se utiliza para cambiar el identificador de las CASIO task.

Describa la precedencia de prioridades para las políticas EDF, RT y CFS, de acuerdo con los cambios realizados hasta ahora.

EDF utiliza las deadlines de las tareas para establecer las prioridades, como su nombre lo indica le da prioridad a los procesos que tienen un deadline pronto. CFS usa el árbol de decisiones red-black para organizar los procesos por prioridad, además balancea las tareas entre los CPU's. Este balanceo esta en Kernel/sched_fair.c, por otro lado RT tiene como objetivo de balanceo estar seguro que N es la tarea con mayor prioridad en el sistema y que se esta ejecutando. En RT un proceso con prioridad menor se corre en otro CPU si: el proceso de menor prioridad ingresa a ejecutarse en un CPU con un proceso de mayor prioridad o si un proceso de mayor prioridad ingresa en un CPU con procesos de menor prioridad.

Explique el contenido de la estructura casio_task

Una casio task representa un proceso, es decir, un nodo en el red-black tree. Su estructura esta compuesta por 4 elementos, un struct rb_node, una variable unsigned long long que representa el deadline del proceso, un struct list_head y un struct task_struct* que es el puntero hacia la tarea.

Explique el propósito de la estructura casio_rq

Esta estructura es como el manejador de las casio_task, se guarda en una lista entrelazada.

¿Qué indica el campo .next de esta estructura?

El .next indica que ahí se guarda la dirección de memoria de la variable rt_sched_class

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?

Una casio_task se almacena en la lista encadenada de la estructura casio_rq que tiene un puntero hacia la tarea. Se actualiza el deadline y se inserta la casio_task en el red-black tree. Se utilizn las funciones de enqueue y dequeue en el momento de añadir y eliminar la tarea que se esta corriendo. La casio_task que contenga el menor tiempo de deadline se selcciona para ser ejecutada.

La casio_task utilizan las estructuras de lista encadenada y red-black tree porque con la lista encadenada pueden iterarse y comunicarse entre sí mientas que el red-black tree es una estructura de balanceo donde los valores solo pueden tomar 2 posiciones y ayuda a las referencias con punteros.

¿Cuándo preemptea una casio_task a la task actualmente en ejecución?

La tarea actual debe ser preempted si hay al menos una caasio_task en la cola de ejecución y la tarea actualmente asignada al procesador no es una casio_task y también si la tarea actualmente en ejecución es una casio_task y hay al menos una casio_task en el red-black tree con una menor deadline.

¿Qué información contiene el archivo system que se especifica como argumento en la ejecución de casio_system?

Es el archivo en donde están definidas las funciones que ayudan a que se ejecuten los casio_task.

Investigue el concepto de aislamiento temporal en relación a procesos. Explique cómo el calendarizador SCHED_DEADLINE, introducido en la versión 3.14 del kernel de Linux, añade al algoritmo EDF para lograr aislamiento temporal.

SCHED_DEADLINE esta basado en el algoritmo EDF y el Constant Bandwith Server (CBS) que permiten reservar recursos. Cada Tarea tiene un tiempo de ejecución y un periodo. Con la utilización de SCHED_DEADLINE las tareas declaran de forma independiente sus requisitos de tiempo, en términos de un per-tarea en tiempo de ejecución necesarios todos los per-tarea periodo y el núcleo se les acepta en el scheduler después de una Prueba de calendarización. Cuando una tarea intenta ejecutarse por más tiempo que su presupuesto asignado, el kernel suspenderá esa tarea y aplazará su ejecución hasta su próximo período de activación. Esta propiedad de conservación del scheduler permite proporcionar aislamiento temporal entre las tareas.