Universidad del Valle de Guatemala
Facultad de ingeniería

Laboratorio 5

Mariana David
Estefanía Elvira 20725
José Pablo Monzón 20309
Priscilla Gonzalez 20689

Guatemala 21 de abril del 2023

A. **Funcionamiento y uso de structs**

Es una estructura de datos que se utiliza para el almacenamiento de información. Las matrices permiten definir diversas variables con elementos. Prácticamente las estructuras se utilizan para representar un registro y definir una estructura. También, se utilizan para almacenar y manipular información importante del sistema. Por ejemplo, el kernel de un SO podría utilizar un struct para representar un archivo en el sistema de archivos del SO, y proporcionar funciones que permitan a los programas del usuario abrir, leer y escribir en el archivo.

B. **Propósito y directivas del preprocesador.**

El preprocesador es una parte importante del proceso de compilación que permite una mayor flexibilidad y control sobre el código fuente. Las directivas del preprocesador permiten que el código fuente sea más modular, portátil y fácil de mantener. Además tiene el control de preparar el código fuente para realizar diversas tareas útiles como la inclusión de archivos, definición de constantes, condicionales de compilación, eliminación de comentarios, entre otros**.**

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

El operador * se utiliza para desreferenciar un puntero, es decir, acceder al valor almacenado en la dirección de memoria a la que apunta el puntero. Por otro lado, el operador & se utiliza para obtener la dirección de memoria de una variable.

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

APT y dpkg son herramientas complementarias para la gestión de paquetes en sistemas operativos basados en Debian. APT proporciona una interfaz más amigable para el usuario final y se utiliza para la gestión de paquetes a nivel de sistema, mientras que dpkg se utiliza para la gestión de paquetes a nivel de archivos y configuración de paquetes. En general, se recomienda utilizar APT para la mayoría de las operaciones de gestión de paquetes, ya que proporciona una interfaz más fácil de usar y se encarga de la resolución de dependencias de manera automática. Sin embargo, en algunos casos, dpkg puede ser necesario para tareas más específicas de gestión de paquetes.

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

El propósito de modificar el archivo kernel_dir/include/linux/sched.h es agregar una nueva política de programación llamada SCHED_CASIO con un valor de 6.

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

El propósito de las declaraciones #ifdef es incluir condicionalmente la definición de SCHED_CASIO solo si se habilita la opción CONFIG_SCHED_CASIO_POLICY.

G. ¿Qué es una task en Linux?

Es una unidad de trabajo que puede ser ejecutada de forma independiente por el sistema operativo, y es identificada por un PID único. La gestión de tareas es una parte fundamental del sistema operativo Linux y es esencial para garantizar el correcto funcionamiento del sistema y las aplicaciones en ejecución.

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

La estructura de datos task_struct se utiliza para representar un proceso o hilo en el kernel de Linux. Contiene información sobre el proceso, como su estado, prioridad, política de programación, uso de memoria, descriptores de archivos, etc. El equivalente de task_struct en Windows es la estructura EPROCESS.

I. ¿Qué información contiene sched_param?

La estructura sched_param contiene parámetros de programación para un proceso o hilo, como su política de programación, prioridad y quantum.

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

La función rt_policy se utiliza para verificar si una política de programación determinada es una política de tiempo real o no. La macro unlikely se utiliza para indicar al compilador que la condición dentro de la declaración if es poco probable que sea verdadera, para que pueda optimizar el código en consecuencia.

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

La política de programación EDF programa tareas según su fecha límite absoluta.

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

La precedencia de prioridades para las políticas EDF, RT y CFS depende de los parámetros específicos utilizados en cada política, los cuales no se especifican en los fragmentos de código dados. Sin embargo, en general, la política EDF programa tareas según su fecha límite absoluta, mientras que las políticas RT y CFS programan tareas según su nivel de prioridad.

M. 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?

La función __setscheduler establece la política de programación y la prioridad de un proceso. La declaración switch establece el campo sched_class del proceso en casio_sched_class si la política es SCHED_CASIO. La función sched_setscheduler verifica si la política dada es válida y establece los parámetros de programación del proceso en consecuencia. Si la política es SCHED_CASIO, también establece los campos de fecha límite y casio_id del proceso.

N. Explique el contenido de la estructura casio_task.

La estructura casio_task representa una tarea que se programa según la política SCHED_CASIO. Contiene un nodo de árbol rojo-negro (casio_rb_node) para la inserción en el árbol de programación, una fecha límite absoluta (absolute_deadline) para la tarea, un nodo de lista (casio_list_node) para la inserción en la lista de tareas CASIO y un puntero a la estructura task_struct que representa la tarea real (task).

O. Explique el propósito y contenido de la estructura casio_rq

La estructura casio_rq es la estructura de datos principal de nuestra implementación del programador EDF. Contiene un árbol rojo-negro (casio_rb_root) y una lista enlazada (casio_list_head) para organizar tareas según sus fechas límite.

P. ¿Qué es y para qué sirve el tipo atomic_t? Describa brevemente los conceptos de operaciones RMW (read-modify-write) y mappeo de dispositivos en memoria (MMIO).

El tipo atomic_t es un tipo proporcionado por el kernel para un entero atómico, lo que significa que puede ser accedido y modificado atómicamente por múltiples hilos sin causar condiciones de carrera. Se utiliza en la estructura casio_rq para llevar un registro del número de tareas actualmente en ejecución en la cola de listos.

La operación de lectura-modificación-escritura (RMW) se refiere a una operación que lee el valor de una ubicación de memoria, lo modifica y luego escribe el nuevo valor de vuelta en la memoria, todo como una sola operación atómica. Esta operación se utiliza a menudo en entornos de múltiples hilos para evitar condiciones de carrera.

La E/S de memoria mapeada (MMIO) se refiere a un mecanismo que permite acceder a dispositivos de hardware como si fueran ubicaciones de memoria. La técnica MMIO

utiliza registros de memoria mapeada para comunicarse con el dispositivo. De esta manera, el dispositivo puede ser controlado mediante la lectura y escritura de los registros de memoria mapeada en lugar de acceder a él a través de instrucciones de E/S específicas, lo que facilita la comunicación del dispositivo para los programadores.

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

El campo "next" en la estructura sched_class indica la siguiente clase de programación en la jerarquía de programación.

R. 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?

El ciclo de vida de casio_task comienza cuando se le asigna la política de programación casio a través de sched_setscheduler. Se agrega a la cola de listos casio utilizando la función enqueue_task_casio, que lo agrega tanto a la lista enlazada como al árbol rojo-negro. Se utiliza la función earliest_deadline_casio_task_rb_tree para encontrar la tarea con la fecha límite más cercana, que luego es seleccionada para su ejecución por pick_next_task_casio. Cuando se desencola una casio_task de la cola de listos utilizando la función dequeue_task_casio, se elimina tanto de la lista enlazada como del árbol rojo-negro. Si la tarea está en un estado ZOMBIE, se elimina permanentemente de la lista enlazada. Las casio_tasks se almacenan tanto en la lista enlazada como en el árbol rojo-negro para permitir búsquedas eficientes en ambas estructuras de datos.

Las casio_tasks se almacenan tanto en la lista enlazada como en el árbol rojo-negro para permitir búsquedas eficientes en ambas estructuras de datos. La lista enlazada se utiliza para mantener un orden secuencial de las tareas y para facilitar la inserción y eliminación. El árbol rojo-negro se utiliza para permitir búsquedas más rápidas de la tarea con la fecha límite más cercana.

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

Una casio_task interrumpe la tarea actualmente en ejecución cuando su fecha límite absoluta es anterior a la fecha límite absoluta de la tarea actualmente en ejecución.

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

Este archivo contiene información de configuración para las tareas que se prueban. Cuenta con argumentos que incluyen la cantidad de procesos asignados por la simulación y parámetros de tiempo que representan la deadline de las tareas ejecutadas.

U. 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.

El aislamiento temporal es la capacidad de garantizar que los procesos en tiempo real tengan tiempos de ejecución predecibles y que no interfieran entre sí. El programador SCHED_DEADLINE en el kernel de Linux versión 3.14 y posteriores mejora el algoritmo EDF para lograr el aislamiento temporal.

SCHED_DEADLINE utiliza un enfoque de programación jerárquica para proporcionar aislamiento temporal. En el nivel más alto, el programador selecciona la tarea de mayor prioridad que tiene una fecha límite anterior a cualquier otra tarea. La tarea seleccionada se le asigna una tajada de tiempo fija, que es el tiempo restante hasta su fecha límite. Si no hay tareas con una fecha límite anterior, el programador selecciona la tarea de mayor prioridad con una fecha límite vencida.

En el segundo nivel, cada tarea se le asigna un presupuesto que especifica la cantidad máxima de tiempo que puede consumir durante un período de programación. El período de programación es igual a la fecha límite, y el presupuesto se restablece al comienzo de cada período de programación.

Si una tarea supera su presupuesto, se degrada a un nivel de prioridad más bajo, donde compite con otras tareas que han excedido sus presupuestos. El nivel de prioridad se determina por la cantidad de tiempo que la tarea ha excedido su presupuesto, con tareas que han excedido sus presupuestos durante períodos más largos teniendo una prioridad más baja.

En general, el programador SCHED_DEADLINE asegura el aislamiento temporal mediante la aplicación de fechas límite y presupuestos para cada tarea en tiempo real, y asignando tajadas de tiempo fijas a las tareas según sus fechas límite. Esto permite al programador tomar decisiones de programación que aseguran que ninguna tarea supere su presupuesto o fecha límite, y que cada tarea tenga un tiempo de ejecución predecible.