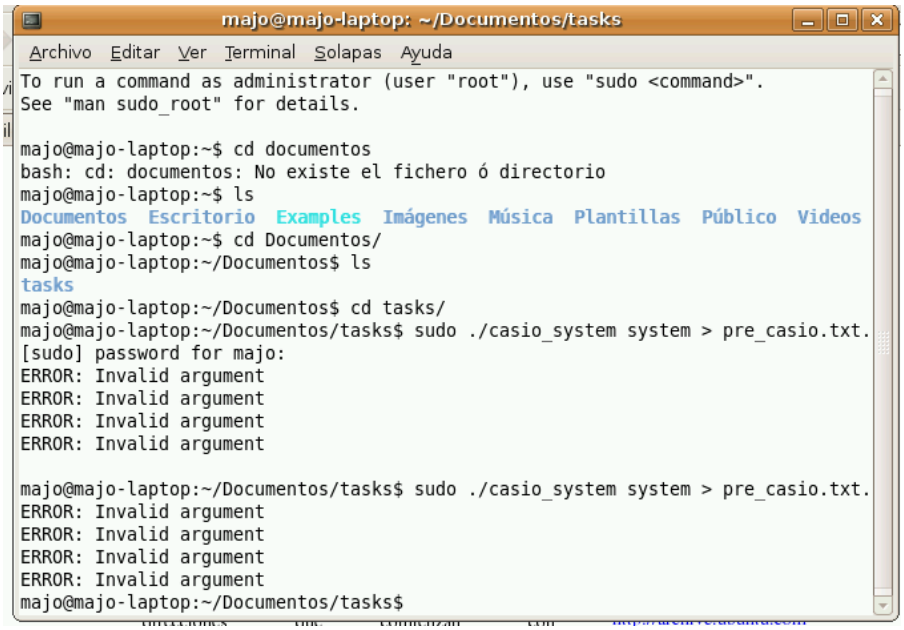
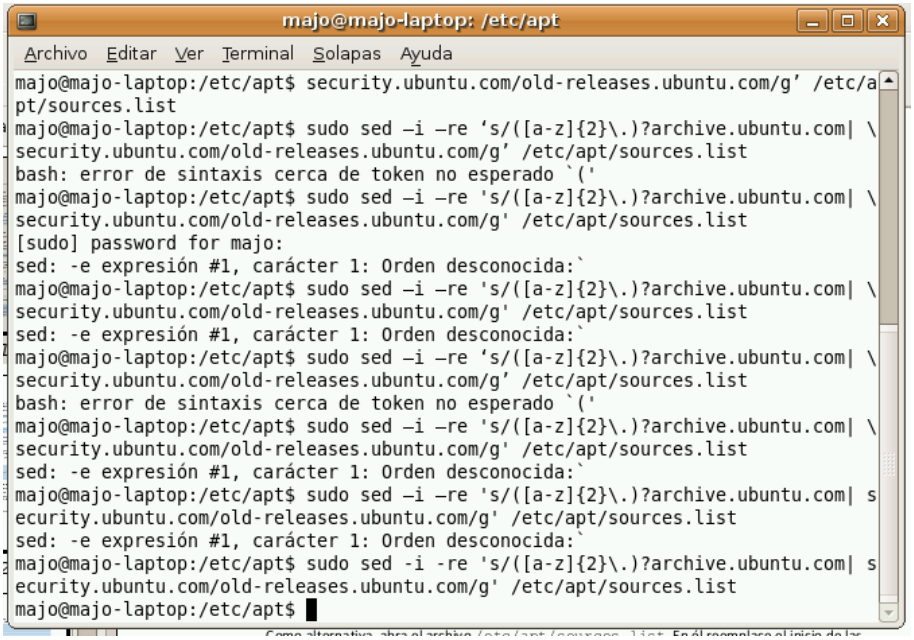
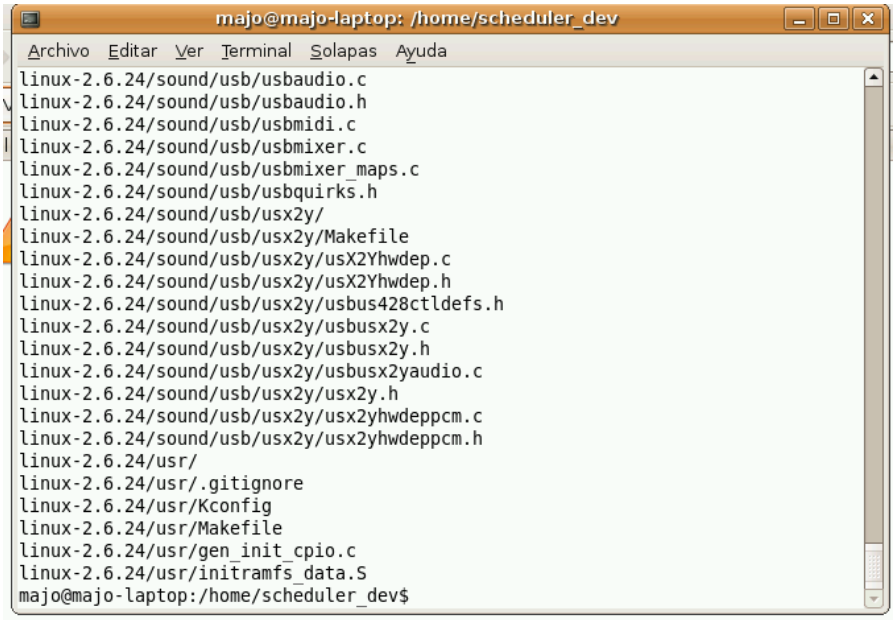
UVG
UNIVERSIDAD
DEL VALLE
DE GUATEMALA

Laboratorio 5
Sistemas Operativos
Maria Jose Castro #181202







1. Funcionamiento y sintaxis de uso de structs.
    1. Propósito y directivas del preprocesador.

El CPP es el procesador para C. Es el primer programa invocado por el compilador y procesa directivas como:

- #if
- #elif
- #else
- #define
- #undef
- #warning
- #region
- #error
- #line
- #endregion
- #pragma
- #pragma warning
- #pragma checksum

Éstas son las encargadas de ver la información del documento antes de completar la compilación verificando que no hayan errores.

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

En este se puede obtener directamente en la dirección de memoria de cualquier variable. Esto se logra al usar el operador unitario &.
A diferencia del operador unitario * se utiliza para obtener lo apuntado por un puntero.

3. Propósito y modo de uso de APT y dpkg.

APT: Éste está basado en una biblioteca que contiene la aplicación central apt fue la primera interfaz desarrollada dentro de un proyecto. A ésta se le puede agregar o eliminar paquetes del sistema, para poder utilizarlo es necesario actualizar la lista de paquetes por medio de apt-update.

dpkg: Éste es un manejador de paquetes de Debian de nivel medio. Es usado para instalar, borrar o gestionar paquetes de Debian/ Linux. Se llama con parámetros desde una línea de comando especificando una acción o más opciones. La acción le dice que hacer y las opciones controlan su comportamiento. También puede ser utilizado como interfaz de dpkg-deb.
Las opciones que nos brinda son:

- build
- content
- info
- field
- control
- extract
- tarfile
- vextract

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

En este caso se definen las diferentes clases de procesos que existen. Al modificarlo se define un macro para identificar la política de calendarización, es por eso que se modifican ambos archivos.

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

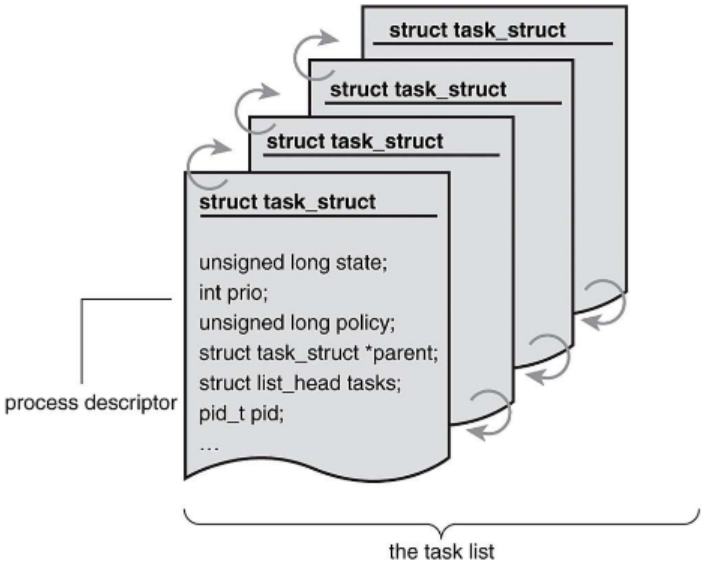
- SCHED_NORMAL (SCHED_OTHER): son las tareas normales realizadas por el usuario (por defecto).
- SCHED_FIFO: las tareas ejecutadas nunca se anularan. Salen del CPU solo para esperar eventos del kernel de sincronización, si se ha solicitado una suspensión explícita o una reprogramación desde el espacio del usuario.
- SCHED_RR: estas tareas son ejecutadas en tiempo real, pero dejarán la CPU si hay otra tarea en tiempo real en la cola de ejecución. Por lo tanto, la potencia del CPU se distribuirá entre todas las tareas de SCHED_RR. Si al menos una tarea de tiempo real se está ejecutando, ninguna otra tarea de SCHED_NORMAL podrá ejecutarse.
- SCHED_BATCH: es una variante del SCHED_IDLE, en donde se podrá hacer uso del CPU solo si los procesos en SCHED_NORMAL no necesitan usarlo.
- SCHED_IDLE: utilizado para ejecutar trabajos de background de muy baja prioridad.
- SCHED_CASIO: nueva política añadida para admitir tareas en tiempo real programadas de acuerdo con el algoritmo de programación de EDF.

6. ¿Qué es una task en Linux?

Particularmente en Linux todo es considerado una tarea. Todo lo que se intercambia entre el CPU y la entidad de tareas task_entity, es una tarea programada. Éstas son programadas a través del comando del kernel Schedule() y pick_up_next_task().

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

Éste es un elemento en la lista de las tareas. Contiene toda la información sobre un proceso especificado. Los procesos en Linux son implementados en el kernel como instancias de tipo task_struct. El campo mm en el task_struct apuntan al memory_descriptor el cual es un resumen ejecutivo de la memoria en el programa. A diferencia de Windows el bloque de et Process es una mezcla entre tasks Track Times Track.



8. ¿Qué información contiene sched_param?

Esta estructura describe los parámetros de calendarización.

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

Esta función determina la política de calendarizador que se dará lugar. La llamada Unlike es la sugerencia para el compilador emita las instrucciones que favorezcan el lado improbable del condicional.

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

Con esta política sabemos que siempre que ocurre un evento de programación (tareas finalizadas nueva tarea liberada etc.) se buscará dentro de la cola el proceso más cercano a una fecha límite. Este proceso es el siguiente en ser programado para ejecutarse y el tipo de tareas que se calendariza son las que maneja Casio.

11. Explique el contenido de la estructura casio_task.

Éste contiene estructura para los nuevos del árbol Red-Black, la etiqueta de los nuevos que es absolute_deadlines y otras dos estructuras correspondiente a los nodos recorridos y las tareas que este maneja.

```
struct casio_task{
        struct rb_node casio_rb_node;
        unsigned long long absolute_deadline;
        struct list_head casio_list_node;
        struct task_struct* task;
};
```

12. Explique el propósito y contenido de la estructura casio_rq.

El propósito de este es coordinar los procesos por prioridad de forma que se tiene una lista con los procesos de forma ascendente la cual será recorrida y ordenada constantemente.

```
struct casio_rq{
        struct rb_root casio_rb_root;
        struct list_head casio_list_head;
        atomic_t nr_running;
};
```

13. ¿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).

Atomic_T es una variable de tipo int que cuenta con la junta operaciones que garantizan ser atómicas sin bloqueo explícito

RMW es una clase de operadores atómicos que leen la ubicación de la memoria y escriben nuevos valores de forma simultánea. Este tipo de operadores evitan las race-conditions.

MMIO y o es una forma de direccionamiento de memoria que permite construir el sistema de manera que el CPU use los registros del chip como si fueran posiciones de memoria RAM.

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

En este caso lo que nos indica es la dirección de memoria de la variable. Se utiliza para organizar los módulos de planificador por prioridad en una lista y al núcleo del calendarizador, iniciando con el módulo del calendarizador de mayor prioridad. Buscará la tarea ejecutable de cada módulo en un orden decreciente según su prioridad.

15. ¿Por qué se guardan las casio_tasks en un red-black tree y en una lista encadenada?

En enqueue_task_casio es llamado cada vez que una Casio_Task que entra en un estado de ejecución. La función invocada Find_Casio, luego lleva el puntero a la tarea estruct_Casio en la lista vinculada. Después, actualiza la fecha límite absoluta e inserta Casio_Task en el árbol Red-Black.

Las tareas son guardadas en el árbol debido a que están tienen que ser ordenadas dependiendo del tiempo. Esto es la prioridad por CFS.

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

Esto se da en el momento de llamar a la función Pick_Next_Casio. Esta función selecciona la tarea que ejecutará el proceso actual. La función es invocada por el núcleo del planificador siempre que la tarea que se está ejecutando se anule. Si no hay ninguna tarea que necesita ser ejecutada a la función actual de Velvet un Núñez y el planificador intenta encontrar otra tarea en la lista, pero de menor prioridad.

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

Éste contiene la duración de las tareas.

18. 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.

En este caso el aislamiento de procesos es un mecanismo para ejecutar programas con seguridad y de forma separada. A menudo es utilizado para ejecutar código nuevo o software de dudosa procedencia, el aislamiento permite controlar de cerca los recursos proporcionados a los programas "cliente" a ejecutarse, tales como espacio temporal de disco y memoria.

SCHED_DEADLINE está basado en el algoritmo Edit Deadline First, el cual reserva recursos. Con este algoritmo en las tareas se declaran como independientes a sus requisitos de tiempo, en términos de un tiempo de ejecución que necesite por tarea, y el Kernel los aceptan en el calendarizador después de una prueba de calendarización. En el caso de que una tarea intente ejecutarse por más tiempo del asignado, el carnet suspenderá la tarea y deberá aplazar su ejecución hasta su próximo periodo de activación. A esto se le conoce como el aislamiento temporal entre tareas.