**EJERCICIOS TEMA 2 SISTEMAS OPERATIVOS.**

1 – Se creará un proceso hijo con los mismos archivos abiertos que su padre, la misma información de SA, pertenecerá al mismo grupo que su padre y compartirá las mismas direcciones de usuario.

Descriptores de tareas:

2 – En primer lugar, dup\_task\_struct copia el descriptor de proceso actual (task\_struct, pila y thread\_info)

Alloc\_pid le asigna un nuevo PID al proceso.

Al iniciar la task\_struct diferenciamos los hilos padre e hijos y ponemos este último en estado no-interrumpible.

Sched\_fork: marcamos el nuevo hilo como task\_running y se inicializa la información sobre la planificación a aplicar.

Copiamos/Compartimos los componentes según los indicadores de la llamada a la función clone()

Asignamos ID, relaciones, etc…

3 – El algoritmo CFS asigna proporciones del procesador en función de la carga del sistema, por tanto cada proceso se ejecutará en un tiempo proporcional a su propio peso, dividido entre la suma de todos los pesos de los procesos actuales, además un proceso de E/S tiene un cpusage menor que un proceso de cpu, por tanto el algoritmo CFS le asigna más CPU para que se mantenga equilibrado.

4 –

1. Las clases que encontramos en un sistema Linux son SCHED\_FIFO, SCHED\_RR y SCHED\_NORMAL.

SCHED\_FIFO: Es el algoritmo más sencillo. Consiste básicamente en que el primer proceso en solicitar acceso a la CPU, es el primero en recibir dicha asignación. La implementación de FCFS se realiza a través de una cola FIFO. Cuando un proceso entra en la cola, su PCB se enlaza con el final de la misma. FCFS funciona mejor con procesos largos, sin embargo sus prestaciones son algo pobres y presenta diversos problemas como: un tiempo de espera elevado, un bajo uso de la CPU, un tiempo de respuesta pobre, uso ineficiente de E/S…

SCHED\_RR: Está diseñado para sistemas de tiempo compartido. En él se define una pequeña unidad de tiempo denominada quantum o time slice y la cola de listos se trata de forma circular. Entonces el planificador recorre esta lista asignándole a cada proceso un intervalo de tiempo del tamaño de un quantum. El problema es que si el quantum es muy pequeño el proceso se moverá por el sistema muy rápido pero deben evitarse estos quantums de tiempo reducido. La mejor idea es que el quantum debe ser ligeramente mayor que el tiempo requerido para la función típica del proceso. Si es menor, es posible que algunos procesos necesiten al menos dos quantums de tiempo. Y por otro lado, si el quantum asignado es muy grande, la planificación degenerará en FCFS. Una desventaja es que trata de forma desigual a los procesos limitados por procesador y los procesos limitados por E/S, estos últimos tienen ráfagas de procesador más cortas y por tanto si hay una mezcla de ambos procesos sucederá que un proceso limitado por E/S usará poco tiempo el procesador y luego se bloqueará esperando que se complete la operación E/S y luego pasa a la cola de listos, mientras que un proceso limitado por procesador utilizará la sección de tiempo asignada de forma completa mientras se ejecuta y posteriormente pasa a la cola de listos, por lo que tienden a recibir rodajas de tiempo poco equitativa, lo que conlleva un mal rendimiento de los procesos limitados por E/S y un uso ineficiente de estos.

Una solución de esto es el uso de Virtual Round Robin, con las mismas características solo que se añade una cola auxiliar a la que se mueven los procesos que se bloquean en una E/S. Así, cuando se toma la decisión de qué proceso activar, los de esta cola auxiliar tienen prioridad sobre los de la cola de listos.

Al ejecutar un proceso desde esta cola auxiliar, se ejecuta por un tiempo no superior a una rodaja de tiempo asignada, menos el tiempo total que ha estado ejecutando desde que no está en la cola de listos.

SCHED\_NORMAL: esta clase se divide en dos:

SCHED\_BATCH: Son los procesos que se ejecutan cuando el procesador se encuentra inactivo, son procesos convencionales.

SCHED\_IDLE: Se trata de procesos con una prioridad muy baja.

b)

La prioridad seria:

Procesos tiempo – real > Procesos CFS > Procesos IDLE

c)

5 – 1º En primer lugar se selecciona la cola de procesos

2º En segundo lugar, desactivamos la tarea actual

3º Posteriormente se selecciona el siguiente proceso a ejecutar

4º Tras ello se realiza el cambio de contexto

5º Finalmente se comprueba si hay que replanificar

Hay que tener en cuenta que todo esto se realiza bajo la lista de prioridades

Tiempo real (fifo o rr) > cfs > idle

6 – Los gobernadores de Linux son políticas utilizadas para ajustar las frecuencias del procesador.

A nivel de usuario disponemos de performance para mantener la cpu a la mayor frecuencia posible y powersave que mantiene la cpu a la menor frecuencia posible. Ambos recurren a Userspace que exporta la información disponible de la frecuencia a nivel de usuario permitiendo el control de la misma.

7 – Son puntos de flujo de ejecución kernel donde es posible apropiar el proceso actual sin incurrir en una condición de carrera.

1. Dado que la invocación asíncrona del planificador puede generar condiciones de carrera, esta se difiere hasta alcanzar un punto de apropiación principal. El estado siempre es consistente, no puedes apropiarlo cuando quieras.

Por otro lado los tiempos de respuesta son más lentos.

8 – a) Suministran un mecanismo para asignar / limitar / priorizar recursos de CPU, memoria y dispositivos; medir el uso de recursos; aislar namespaces separados por grupos; congelar grupos o hacer checkpoint/restar de estos.

1. Dan soporte a la virtualización de la forma comentada anteriormente
2. freezer, ns, devices, blkio, cpuacct, cpu, net\_cls, memory, cpust