INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

INGENIERIA EN SISTEMAS COMPUTACIONALES

MATERIA: SISTEMAS OPERATIVOS

PROFESOR: ARAUJO DIAZ DAVID

PRESENTA:

RAMIREZ BENITEZ BRAYAN

NÚMERO DE LISTA:

27

GRUPO: 2CV17

TAREA 2:

ADMINISTRACIÓN DE PROCESOS

CIUDAD DE MEXICO MARZO DE 2021

1. ¿Qué es el seudoparalelismo?

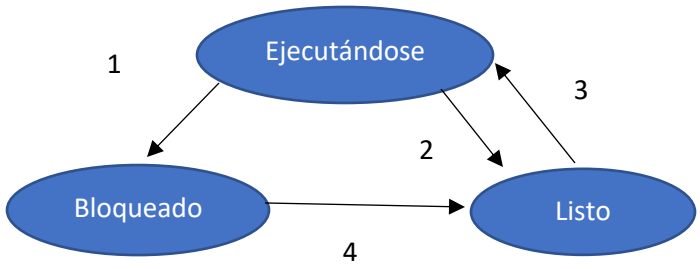
R: Es la rápida conmutación del CPU entre programas.

2. Describa en que consiste la jerarquía de procesos.

R: Es el mecanismo que crea y destruye procesos según sea necesario durante la operación.

3. ¿Cuáles son los tres estados en los que se puede encontrar un proceso y cuáles son sus transiciones?

R: Ejecutándose, Listo y Bloquead.



1. Un proceso se bloquea para aceptar entradas.
2. El planificador escoge otro proceso.
3. El planificador escoge este proceso.
4. Hay entradas disponibles.

4. Describa como se realiza la implementación de procesos, mediante una tabla de procesos y vectores de interrupción.

R:

1. El hardware agrega a la pila el contador de programa
2. El hardware carga un nuevo contador de programa del vector de interrupciones
3. El procedimiento en lenguaje ensamblador guarda los registros

4. El procedimiento en lenguaje ensamblador prepara una nueva pila
5. Se ejecuta el servicio de interrupciones en lenguaje C
6. El planificador marca la tarea en espera como lista
7. El planificador decide cual proceso debe ejecutarse a continuación
8. El procedimiento en lenguaje C regresa al código en ensamblador
9. El procedimiento en lenguaje ensamblador inicia el nuevo proceso actual

5. ¿Cuál es la diferencia entre un hilo y un proceso?

R: Un proceso es una entidad de ejecución independiente, donde, el sistema operativo, en el momento en que el proceso se lanza, proporciona un espacio de direcciones de memoria en los que el proceso puede ejecutarse, mientras que los hilos son entidades de ejecución independiente que viven dentro de los procesos y, por tanto, viven dentro del mismo espacio de direcciones de memoria que otros hilos, lo que permite acceder a cualquier dato dentro del mismo proceso.

6. ¿Qué son los hilos?

R: Es una secuencia de tareas encadenadas muy pequeña que puede ser ejecutada por un sistema operativo.

7. Liste los estados en los que se puede encontrar un hilo.

R:
- Ejecutándose
- Listo
- Bloqueado

8. ¿Por qué es necesaria la comunicación entre procesos?

R: Para enviar y recibir información, asegurarse de que dos o mas procesos no se estorben mutuamente al efectuar actividades y para una secuencia correcta cuando existen dependencias.

9. Defina que es una condición de competencia.

R: Ocurre cuando dos o más procesos acceden un recurso compartido sin control, de manera que el resultado combinado de este acceso depende del orden de llegada.

10. ¿Qué son las secciones críticas?

R: La porción de código de un programa de ordenador en la que se accede a un recurso compartido que no debe ser accedido por más de un proceso o hilo en ejecución.

11. Mencione las cuatro condiciones para evitar condiciones de competencia, empleando secciones críticas.

R: 1. Dos procesos nunca pueden estar simultáneamente dentro de sus regiones críticas.

2. No pueden suponerse nada acerca de las velocidades o el número de las CPU.

3. Ningún proceso que se ejecute fuera de su región critica puede bloquear a otros procesos.

4. Ningún proceso deberá tener que esperar indefinidamente para entrar en su región critica.

12. Describa la exclusión mutua con espera activa y describa:

a. Inhabilitación de interrupciones

R: Consiste en hacer que cada proceso inhabilite las interrupciones justo después de ingresar en su región critica y vuelva a habilitarlas justo antes de salir de ella, con las interrupciones inhabilitadas, no pueden ocurrir interrupciones de reloj.

b. Variables candado

R: Cuando un proceso quiere entrar en su región critica, lo primero que hace es probar el candado. Si el candado es 0, el proceso le asigna 1 y entra en su región critica; si es 1, el proceso espera hasta que el candado vuelve a ser 0. Así, un 0

significa que ningún proceso está en su región critica, y un 1 significa que algún proceso está en su región critica.

c. Alternancia estricta

R: Requiere que los dos procesos se alternen de manera estricta al entrar en sus regiones críticas. Ninguno podría poner dos archivos en la cola al mismo tiempo. Aunque este algoritmo evita todas las condiciones de carrera, en realidad no es un candidato serio como solución.

13. Describa las soluciones para la exclusión mutua de:

a. Dekker

R:

```
shared int cierto = 1;

/* ''Definición de variables compartidas'' */
shared int bandera[2] = {0,0};
shared int turno      = 0;

while (cierto)
{
   bandera[proc_id] = cierto;  // Está listo para acceder a la Sección Crítica
   while (bandera[1-proc_id] == cierto) { // Mientras el otro esté procesando
      if (turno == 1-proc_id) {           // si es su turno
         bandera[proc_id] = 0;            // indicar que no está listo y
         while (turno == (1-proc_id)) {}  // esperar activamente.
         bandera[proc_id] = 1;            // Cuando terminó de esperar, está listo
      }
   }
   /* ''Sección crítica'' */
   turno = 1-proc_id; // Indicar el turno del otro proceso
   bandera[proc_id] = 0;  // Indicar que ya no está listo (para acceder a la Sección Crítica)
   /* ''Sección no crítica'' */
}
```

b. Paterson

R:

```
#define FALSE 0
#define TRUE 1
#define N  2                                    /* número de procesos */

int turno;                                      /* ¿de quién es el turno? */
int interesado[N];                              /* al principio todos los valores son 0 (FALSE) */

void entrar_region(int proceso);                /* el proceso es 0 o 1 */
{
    int otro;                                   /* número del otro proceso */

    otro = 1 – proceso;                         /* el opuesto del proceso */
    interesado[proceso] = TRUE;                 /* muestra que está interesado */
    turno = proceso;                            /* establece la bandera */
    while (turno == proceso && interesado[otro] == TRUE)  /* instrucción nula */;
}

void salir_region(int proceso)                  /* proceso: quién está saliendo */
{
    interesado[proceso] = FALSE;                /* indica que salió de la región crítica */
}
```

c. TSL (Test and Set Lock)

R:

enter_region:
ts1 register.flac copia fag al registro *y* hace fag=1;
cr.r register.#0 ¿fag=0?
jn.í enter_region: si era distinto de cero , la cerradura estaba establecida, por lo que hay que hacer un ciclo
reg regresa a quien hizo la llamada
entra a la region critica
leave_region:
mov *fag.#0* almacena un 0 en flag
ret regresa a quien hizo la llamada

d. Dormir y despertar

R: - Dormir (sleep): llamada al sistema que provoca el bloqueo del proceso que hizo la llamada (que será suspendido hasta que otro proceso lo despierte).

- Despertar (Wakeup): tiene un parámetro que es el proceso por despertar.

Los procesos se pueden comunicar entre sí mediante las primitivas de comunicación entre procesos, que se utilizan para garantizar que dos procesos no se encuentren jamás al mismo tiempo dentro de sus regiones críticas, es decir la excusión mutua.

14. En que consiste el problema del productor consumidor y muestre como se resuelve con las primitivas SLEEP y WAKEUP.

R: Son aquellos problemas en los que existe un conjunto de procesos que producen información que otros procesos consumen, siendo diferentes las velocidades de producción y consumo de la información.

Solución: añadir un bit de espera por la señal que despierta al proceso (wakeup waiting bit). Este bit se activa cuando se envía una señal para despertar a un proceso que todavía está despierto. Posteriormente, cuando el proceso intente dormirse, si encuentra ese bit activo, simplemente lo desactiva permaneciendo despierto.

15. ¿Qué es un semáforo y como resuelve el problema del productor consumidor?

R: Permiten resolver la mayoría de los problemas de sincronización entre procesos y forman parte del diseño de muchos sistemas operativos y de lenguajes de programación concurrentes. La idea es utilizar una variable entera para contar el número de señales enviadas para despertar un proceso guardadas para su uso futuro. El valor de un semáforo puede ser 0, indicando que no se ha guardado ninguna señal, o algún valor positivo de acuerdo con el número de señales pendientes para despertar al proceso. Operaciones sobre los semáforos: down y up(las cuales generalizan las operaciones sleep y wakeup, respectivamente).

```
tuberia.full.down();          tuberia.empty.down();
tuberia.mutex.down();         tuberia.mutex.down();
c = tuberia.consumir();       tuberia.producir(c);
tuberia.mutex.up();           tuberia.mutex.up();
tuberia.empty.up();           tuberia.full.up();
```

16. ¿Qué es un monitor y como resuelve el problema del productor consumidor?

R: Es una colección de procedimientos, variables y estructuras de datos que están todos agrupados juntos en un tipo especial de módulo o paquete. Los procesos pueden llamar a los procedimientos de un monitor siempre que quieran, pero no se les permite acceder directamente a las estructuras de datos internas del monitor desde procedimientos declarados fuera del monitor. En cualquier instante solamente un proceso puede estar activo dentro del monitor. Los monitores son construcciones del lenguaje, por lo que el compilador sabe que son especiales, de manera que puede tratar las llamadas a los procedimientos del monitor de forma diferente que a otras llamadas a procedimientos normales. Si otro proceso está actualmente activo dentro del monitor, el proceso que hizo la llamada debe suspenderse hasta que el otro proceso abandone el monitor. Si ningún otro proceso está utilizando el monitor, el proceso que hizo la llamada puede entrar inmediatamente.

17. Describa los siguientes problemas clásicos de la comunicación entre procesos:

a. Problema de los lectores escritores

R: Es un dilema de programación creado cuando varios lectores y escritores necesitan acceder al mismo recurso. Si se les permitiera el acceso a todos a la vez, podrían surgir problemas como sobrescripciones, información incompleta y otros problemas. Por lo tanto, los programadores pueden restringir el acceso para

controlar qué subprocesos de procesamiento ven el recurso y cuándo, considerando las necesidades del sistema y los usuarios. Hay varias formas de abordar el problema de lectores escritores. Una de las soluciones más comunes implica el uso de semáforos para marcar el estado y controlar el acceso.

b. Problema de la Cena de los filósofos

R: Es aquel donde varios filósofos sentados en una tabla durante una cena, se dedican a pensar y cuando tienen hambre comen usando para ello dos tenedores que comparten con sus compañeros que se sientan a la izquierda y derecha.

Dado que dos filósofos no pueden utilizar el mismo tenedor a la vez hay que implementar sincronización a la hora de utilizarlos. En la realidad un filósofo representa a un proceso y un tenedor representa a un recurso compartido de uso exclusivo.

c. Problema del peluquero dormido

R: Es un problema de sincronización que consiste en una barbería en la que trabaja un barbero que tiene un único sillón de barbero y varias sillas para esperar. Cuando no hay clientes, el barbero se sienta en una silla y se duerme. Cuando llega un nuevo cliente, éste o bien despierta al barbero o si el barbero está afeitando a otro cliente se sienta en una silla (o se va si todas las sillas están ocupadas por clientes esperando). El problema consiste en realizar la actividad del barbero sin que ocurran condiciones de carrera. La solución implica el uso de semáforos y objetos de exclusión mutua para proteger la sección crítica.

18. ¿Qué es la planificación de procesos?

R: La planificación de procesos es un intercambio de información entre la memoria principal y el disco y determina el tiempo de ejecución de cada proceso. Una complicación que deben asumir los planificadores es que cada proceso se ejecute durante demasiado tiempo. Casi todas las computadoras tienen incorporado un cronómetro o reloj electrónico que genera interrupciones periódicamente.

19. ¿Cuáles son los criterios para tener un buen algoritmo de planificación?

R: •Equidad: Asegurarse de que cada proceso reciba una parte justa de tiempo de CPU.

•Eficacia: Mantener la CPU ocupada todo el tiempo.

•Tiempo de respuesta: Minimizar el tiempo de respuesta para usuarios interactivos.

•Retorno: Minimizar el tiempo que los usuarios por lotes tienen que esperar sus salidas

20. ¿Qué es la planificación expropiativa y la no expropiativa?

R: Planificador expropiativo son aquellos que pueden expropiar el recurso procesador a un proceso cuando otro proceso entra en estado pronto (ya sea porque es nuevo o porque se desbloqueó) o porque se le impone un límite de tiempo para ejecutar.

Planificador no expropiativo son aquellos que asignan el recurso procesador a un proceso y hasta que este no lo libere, ya sea porque finaliza su ejecución o se bloquea, no se vuelve a ejecutar el planificador.

21. Describa los siguientes tipos de planificación de procesos:

a. Planificación Round Robin

R: A cada proceso se le asigna un intervalo de tiempo de ejecución llamado quantum. Si el proceso continúa en ejecución al final de su quantum, otro proceso se apropia de la CPU. Si el proceso está bloqueado o ha terminado antes de consumir su quantum, se alterna el uso de la CPU.

b. Planificación por prioridad

R: A cada proceso se le asigna una prioridad y se permite que se ejecute el proceso que tenga la prioridad más alta. A fin de evitar que los procesos de alta prioridad se ejecuten indefinidamente, el planificador puede reducir la prioridad de los procesos que actualmente se ejecutan cada tic del reloj. Si esta acción hace que la prioridad se vuelva menor que la del siguiente proceso con más alta prioridad, ocurrirá una conmutación de procesos. Como alternativa se podría asignar a cada proceso un

cuanto máximo en que se le permitiera tener la CPU continuamente, cuando se agota ese cuanto, se da oportunidad al proceso con la siguiente prioridad más alta de ejecutarse.

c. Planificación por colas múltiples

R: Se establecen clases de prioridad con diferente cantidad de quantums, los de la clase de mayor prioridad se ejecutan en un quantum, los de la siguiente en dos quantums y así sucesivamente. Cuando un proceso consume todos sus quantums, se lo mueve a la siguiente clase.

d. Planificación por el trabajo más corto

R: Apropiado para las tares por lotes, donde se conocen los tiempos de ejecución previamente. Consiste en utilizar el criterio de ejecutar primero el trabajo más corto, cuando varios trabajos de similar importancia esperan en la lista para iniciar. Tiempo promedio de regreso para cuatro tareas a,b,c,d: para a el tiempo a; para b, a+b; para c, a + b + c; para d, a + b + c + d; con lo cual queda: (4a+3b+2c+d)/4 produce el promedio mínimo de tiempo de respuesta.

e. Planificación garantizada

R: Consiste en hacer promesas reales al usuario y después cumplirlas (en cuanto al rendimiento). El sistema calculará el tiempo de CPU al que tenía derecho cada proceso, es decir, el tiempo de creación dividido la cantidad de usuarios. Puesto que también se conoce el tiempo de CPU del que cada proceso ha disfrutado realmente, es fácil calcular la relación entre el tiempo de CPU recibido y el tiempo al que tenía derecho. El algoritmo consiste en ejecutar el trabajo con la relación más baja hasta que su relación haya rebasado a la de su competidor más cercano

f. Planificación por lotería

R: La idea básica consiste en dar a los procesos boletos de lotería para los diversos recursos del sistema, como el tiempo de CPU: Cada vez que se hace necesario tomar una decisión de planificación, se escoge al azar un boleto de lotería, y el proceso poseedor de este boleto obtiene el recurso.

g. Planificación en tiempo real

R: Dispone de un planificador de procesos que tiene mecanismos para hacer lo máximo posible para garantizar que sus procesos de tiempo real cumplan los plazos de finalización que tienen establecidos.

h. Planificación de dos niveles

R: Si la memoria principal no es suficiente, alguno de los procesos ejecutables tendrá que mantenerse en el disco total o parcialmente. Esta situación tiene implicaciones importantes para la planificación, ya que el tiempo de conmutación de procesos cuando hay que traer procesos del disco es varias órdenes de magnitud mayor que cuando la conmutación es a un proceso que ya está en memoria. Primero se carga en la Memoria Principal un subconjunto de los procesos ejecutables, luego el planificador se limita a escoger procesos de ese subconjunto durante un cierto tiempo.

22. ¿Cómo se planifican los procesos en UNIX?

R: La política de planificación que utiliza este planificador es del tipo round robin con colas multinivel. Cada proceso tiene asignada una prioridad de planificación que cambia con el tiempo. Dicha prioridad le hace pertenecer a una de las múltiples colas de prioridad que maneja el planificador.

El planificador siempre selecciona al proceso que encontrándose en el estado preparado en memoria principal para ser ejecutado o en el estado expropiado tiene la mayor prioridad. En el caso de los procesos de igual prioridad (se encuentran en la misma cola) lo que hace es ceder el uso de la CPU a uno de ellos durante un cuanto, cuando finaliza dicho cuanto le expropia la CPU y se lo cede a otro proceso. El planificador varía dinámicamente la prioridad de los procesos basándose en su tiempo de uso de la CPU. Si un proceso de mayor prioridad alcanza el estado preparado en memoria principal para ser ejecutado, el planificador expropia el uso de la CPU al proceso actual incluso aunque éste no haya completado su cuánto.

El núcleo tradicional de UNIX es estrictamente no expropiable. Es decir, si el proceso actual se encuentra en modo núcleo (debido a una llamada al sistema o a una interrupción), no puede ser forzado a ceder la CPU a un proceso de mayor prioridad.

Dicho proceso cederá voluntariamente la CPU cuando entre en el estado dormido. En caso contrario sólo se le podrá expropiar la CPU cuando retorne a modo usuario.

23. ¿Cuáles son las formas de comunicación entre procesos en UNIX?

R: Las primeras distribuciones de UNIX únicamente disponían de tres mecanismos que podían ser utilizados para la comunicación entre procesos:

- las señales
- las tuberías
- el seguimiento de procesos.

24. ¿Qué es una señal en UNIX y que las provoca?

R: Las señales se utilizan principalmente para notificar a un proceso eventos asíncronos. Originariamente fueron concebidas para el tratamiento de errores, aunque también pueden ser utilizadas como mecanismo IPC. Las versiones modernas de UNIX reconocen más de 31 señales diferentes. La mayoría tienen un significado predefinido, pero existen dos, SIGUSR1 y SIGUSR2, que pueden ser utilizadas por los usuarios según sus necesidades. Un proceso puede enviar una señal a un proceso o a un grupo de procesos usando por ejemplo la llamada al sistema kill. Además, el núcleo genera señales internamente en respuesta de distintos eventos.

25. ¿Qué son los candados en UNIX?

R: El candado es una estructura de datos que posee dos operaciones:

- Lock(&lock): Si el candado está cerrado espera hasta que esté abierto. Cuando el candado está abierto, cierra el candado y retorna de inmediato.
- Unlock(&lock): Abre el candado y retorna de inmediato.

El candado se cierra al principio de la rutina de atención de las interrupciones por llamadas al sistema. Es decir, cuando el procesador ya se encuentra en el modo sistema, pero todavía no ha realizado ninguna acción peligrosa. Si el procesador encuentra cerrado el candado, se queda bloqueado hasta que el candado sea abierto por el procesador que se encuentra en el modo sistema. Mientras tanto el procesador bloqueado no puede ejecutar otros procesos, pues dado que la cola ready es una estructura de datos compartida, no puede tener acceso a ella para

extraer y retomar un proceso. El candado se abre en la rutina de atención justo antes de retornar a la aplicación.

26. ¿Mencione algunos ejemplos de filtros en UNIX?

R: “sort**”:** Muestra por la salida estándar el contenido de un fichero ordenado de manera alfabética. Se pueden utilizar los siguientes parámetros:

- -n: emplear orden numérico.
- -d: emplear orden alfabético (por defecto).
- -f: ignorar mayúsculas y minúsculas.
- -r: ordenación inversa.
- -k: ordenar por un determinado campo.
- -t: definir el carácter de separación de campo (espacio por defecto).

```
sort doc1
sort –r doc1
```

“Shuf**”:** Muestra por la salida estándar las líneas del contenido de un fichero de manera aleatoria:

```
shuf doc1
```

“More**”:** Muestra por la salida estándar el contenido de un fichero, pero paginado. Similar a la orden cat pero página a página y no entero de una sola vez.

```
more doc1
```

Referencias

Implementación de procesos en Unix. (s. f.). Implementacion de procesos en Unix. Recuperado 14 de marzo de 2021, de https://users.dcc.uchile.cl/%7Ejpiquer/Docencia/SO/aps/node24.html

OCW UPM. (s. f.). *tema_6_03 — OCW UNED*. OCW UNED. Recuperado 14 de marzo de 2021, de http://ocw.innova.uned.es/ocwuniversia/Ing_tecnico_infor_sistemas/SO_II/contenidos_html/ims_import_5/tema_6_03.htm

Sistemas en Tiempo Real. (s. f.). Sistemas en Tiempo Real. Recuperado 14 de marzo de 2021, de https://www.dsi.fceia.unr.edu.ar/images/downloads/informatica/info_III/Concurrencia2-2014.pdf

Sistemas Operativos. (s. f.). Sistemas Operativos. Recuperado 14 de marzo de 2021, de https://www.fing.edu.uy/inco/cursos/sistoper/recursosTeoricos/6-SO-Teo-Planificacion.pdf

Sistemas Operativos II. (s. f.). SOPII. Recuperado 14 de marzo de 2021, de https://www.profesores.frc.utn.edu.ar/sistemas/ingsanchez/SOP/Archivos/SOPII.pdf