Practica 4 CSO

1)

a) Un programa es un conjunto de instrucciones estáticas que se almacenan en disco mientras que un proceso es un programa en ejecución, que incluye código, estado, recursos asignados y datos

b)

* TR: tiempo total desde que un Job entra al sistema hasta que finaliza à TR = Tiempo de Fin – Tiempo de Llegada
* TE: tiempo que un Job pasa esperando en la cola de ready (sin ejecución) à TE = TR – Tiempo de Ejecución

c)

* TPR: promedio de los tiempos de retorno de todos los Jobs en un lote
* TPE: promedio de los tiempos de espera de todos los Jobs en un lote

d) El quantum es un intervalo de tiempo fijo asignado a cada proceso en un sistema de Scheduling por Round Robin

e)

* Algoritmo Apropiativo (Preemptive): Permite interrumpir un proceso en ejecución para dar paso a otro mas prioritario
* No Apropiativo (Non-Preemptive): El proceso en ejecución se mantiene hasta completarse o realizar una operación de E/S.

f)

1. Short Term Scheduler: selecciona que proceso pasa de la cola de ready a la CPU.
2. Long Term Scheduler: decude que procesos cargan desde almacenamiento secundario a memoria principal.
3. Medium Term Scheduler: suspende o reanuda procesos para optimizar recursos (swapping)

g) El dispacher asigna la CPU al proceso seleccionado por el short term scheduler. Luego realiza el cambio de contexto y transfiere el control al proceso en ejecución.

2)

a)

* Top: muestra una lista de los procesos que se están ejecutando en tiempo real. También proporciona información sobre el uso de la CPU, memoria, tiempo de ejecución, etc.
* Htop: similar a top pero con una interfaz mas amigable y colorida. Permite interactuar con los procesos mediante teclas para finalizar o cambiar prioridades.
* Ps: muestra los procesos que se están ejecutando en ese instante. Permite filtrar con información especifica como el usuario, PID, estado y mas.
* Pstree: muestra los procesos en ejecución en modo de árbol jerárquico.
* Kill: envía señales a procesos identificados por su PID, como SIGTERM para finalizar el procesos de forma segura o SIGKILL para forzar su finalización.
* Pgrep: Busca procesos por nombre y devuelve los PIDs que coincidan, facilitando su identificación para comandos como kill.
* Killall: Finaliza todos los procesos que coincidan con un nombre específico, en lugar de identificar procesos por **PID** como kill.
* Renice: Cambia la prioridad (nice value) de un proceso en ejecución para ajustar su uso de CPU. Los valores más bajos dan mayor prioridad.
* Xkill: Herramienta gráfica para forzar la finalización de una ventana o proceso asociado haciendo clic sobre ella. Requiere un entorno gráfico.
* Atop: Herramienta avanzada para monitorear procesos y recursos del sistema en tiempo real. Muestra información detallada sobre CPU, memoria, disco, red y procesos específicos, incluyendo su historial.

d)

1. Los pipes son mecanismos de comunicación unidireccionales que permiten la transferencia de datos entre procesos. Un proceso escribe en el extremo de entrada (write-end) y otro proceso lee desde el extremo de salida (read-end).
2. Las pipes en C se crean de la siguiente forma: A screen shot of a computer

   Description automatically generated
3. El parámetro necesario para la creación de una pipe es un array denominado “pipefd”, que es un arreglo de dos enteros que actúa como un descriptor de archivo.
4. Los tipos de comunicación posibles con pipes son:
   1. Unidireccional: los pipes anónimos permiten comunicación de un solo sentido (de un proceso a otro).
   2. Bidireccional (con dos pipes): usando dos pipes independiente, se logra comunicación bidireccional (uno para enviar datos en cada dirección).
   3. Entre procesos relacionados: con pipes anónimos, la comunicación ocurre entre procesos padre e hijo.
   4. Entre procesos no relacionados: usando pipes con nombres (FIFOs), su puede comunicar cualquier proceso en el sistema
5. La información mínima incluye PID, PPID, estado del proceso, contexto de ejecución (registros de la CPU, estado de la pila y memoria asignada), información de memoria, información de recursos asignados, prioridad y tiempo, e información de sincronización.
6. Los procesos “CPU Bound” son procesos que realizan muchas operaciones intensivas en la CPU, pasan la mayor parte de su tiempo ejecutando cálculos o instrucciones complejas. Por otro lado los “I/O Bound” con procesos que dependen principalmente de operaciones de entrada/salida, pasan mas tiempo esperando datos de dispositivos como discos.
7. Estados posibles de un proceso
   1. Nuevo (New): el proceso se está creando
   2. Listo (Ready): el procese esta preparado para ejecutarse, esperando asignación de CPU.
   3. Ejecutando (Running): el proceso esta en ejecución en la CPU.
   4. Bloqueado (Blocked o Waiting): el proceso esta esperando un evento.
   5. Terminado (Terminated): el proceso ha completado su ejecución.
8. Diagrama de transición de estados:
   1. Nuevo à Listo: El proceso esta creado y listo para ejecutarse
   2. Listo à Ejecutando: El scheduler solicita un recurso o realiza E/S
   3. Ejecutando à Bloqueado: el proceso solicita un recurso o realiza E/S
   4. Ejecutando à Listo: el proceso es interrumpido por el scheduler (preemptive)
   5. Bloquedo à Listo: Se cumple el evento esperado (como finalizar E/S)
   6. Ejecutando à Terminado: El proceso finaliza su tarea
9. Diagrama: A diagram of a work flow

   Description automatically generated

3)

Suponiendo la siguiente tabla como ejemplo:

|  |  |  |  |
| --- | --- | --- | --- |
| Proceso | Tiempo de llegada | Ti. De ejecución (CPU burst) | Prioridad |
| P1 | 0 | 5 | 2 |
| P2 | 1 | 3 | 1 |
| P3 | 2 | 8 | 3 |
| P4 | 3 | 6 | 2 |

FCFS (First Come First Served): orden de ejecucion segun orden de llegada

|  |  |  |
| --- | --- | --- |
| Proceso | Tiempo de Espera (TE) | Tiempo de retorno (TR) |
| P1 | 0 | 5 |
| P2 | 4 | 7 |
| P3 | 6 | 14 |
| P4 | 12 | 18 |

SJF (Shortest Job First): orden de ejecucion segun el tiempo de ejecución (CPU burst)

|  |  |  |
| --- | --- | --- |
| Proceso | Tiempo de Espera (TE) | Tiempo de retorno (TR) |
| P1 | 0 | 5 |
| P2 | 4 | 7 |
| P3 | 7 | 13 |
| P4 | 13 | 21 |

Round Robin: Quantum 4 unidades de tiempo

|  |  |  |
| --- | --- | --- |
| Proceso | Tiempo de Espera (TE) | Tiempo de retorno (TR) |
| P1 | 4 | 8 |
| P2 | 7 | 10 |
| P3 | 13 | 21 |
| P4 | 12 | 18 |

Prioridades: orden de ejecución según la prioridad

|  |  |  |
| --- | --- | --- |
| Proceso | Tiempo de Espera (TE) | Tiempo de retorno (TR) |
| P1 | 0 | 3 |
| P2 | 3 | 8 |
| P3 | 8 | 14 |
| P4 | 14 | 22 |

Parámetros necesarios para su funcionamiento:

* FCFS: solo requiere conocer el tiempo de llegada
* SJF: requiere conocer los tiempos de ejecución (CPU burst) de cada proceso
* Round Robin: Necesita un quantum (intervalo fijo de tiempo asignado a cada proceso)
* Prioridades: cada proceso necesita un valor de prioridad asociado

Proceso mas adecuado según el tiempo de SO o procesos:

* FCFS: adecuado para sistemas con pocos procesos y tiempos de ejecución similares, ya que es simple de implementar.
* SJF: ideal para cargas de trabajo predecibles (como sistemas batch), pero no para sistemas interactivos.
* Round Robin: excelente para sistemas multitarea e interactivos, garantiza equidad.
* Prioridades: adecuado para sistemas en tiempo real, donde ciertos procesos son más críticos que otros.

|  |  |  |
| --- | --- | --- |
| Algoritomo | Ventajas | Desventajas |
| FCFS | Simple de implementar | No equitativo, alto tiempo de espera |
| SJF | Minimiza el tiempo promedio de espera | Difícil de predecir tiempos de ejecución |
| Round Robin | Equitativo, buen rendimiento para interactivos | Penaliza procesos largos (overhead por cambio de contexto) |
| Prioridades | Da importancia a procesos críticos | Posible inanición de procesos de baja prioridad |

4)

a) Timer Fijo: el quantum es constante para todos los procesos. Es la implementación estándar de Round Robin y todos los procesos reciben una cantidad igual de tiempo en cada turno

Timer variable: el quantum varia dinámicamente según las características del proceso o la carga del sistema, procesos mas cortos o interactivos pueden recibir menos tiempo, mientras que los procesos mas largos pueden recibir mas para mejorar la eficiencia

b) Suponiendo el siguiente ejemplo:

|  |  |  |
| --- | --- | --- |
| Proceso | Tiempo de llegada | Tiempo de ejecución (CPU Burst) |
| P1 | 0 | 7 |
| P2 | 1 | 4 |
| P3 | 2 | 9 |

Round Robin con timer fijo (quantum = 4):

|  |  |
| --- | --- |
| Tiempo | Ejecución |
| 0-4 | P1 (4) |
| 4-8 | P2 (4 – completo) |
| 8-12 | P3 (4) |
| 12-15 | P1 (3 – completo) |
| 15-18 | P3 (5 – completo) |

Round Robin con timer variable (quantum adaptativo):

|  |  |
| --- | --- |
| Tiempo | Ejecución |
| 0-4 | P1 (4) |
| 4-6 | P2 (2) |
| 6-11 | P3 (5) |
| 11-13 | P2 (2 – completo) |
| 13-15 | P1 (3 – completo) |
| 15-19 | P3 (4 – completo) |

c) La información del quantum en el timer fijo reside en una variable global dentro del kernel, esta es constante para todos los procesos. En el timer variable debe almacenarse en el PCB de cada proceso, esto permite asignar un quantum personalizado a cada proceso y administrarlo de manera dinámica.

9)

a) La inanición es una situación en la que un proceso nunca recibe los recursos necesarios para continuar su ejecución porque otros procesos con mayor prioridad o ventaja siguen acaparando esos recursos.

Ocurre comúnmente en algoritmos de planificación que favorecen criterios como prioridad o tiempo de llegada.

b) Los algoritmos que pueden provocar la inanición son el SJF, el algoritmo basado en prioridades y el Round Robin (no en todos los casos).

c) Las técnicas que evitan la inanición son el aging, que aumenta la prioridad de un proceso a medida que pasa más tiempo esperando. Otro es el control de carga (Load Balancing) que distribuye los procesos de manera equitativa entre recursos para evitar que algunos procesos monopolicen el sistema. También está el quantum ajustado, que en round robin permite ajustarlo de forma dinámica para garantizar que los procesos más largos también progresen de manera justa. Por último, están las restricciones en la ejecución, esto significa que en algoritmos basados en prioridad se puede imponen un límite al número de ejecuciones consecutivas de procesos de alta prioridad antes de atender los de baja prioridad.

11)

Los algoritmos de planificación de procesos en SOs tienen ciertas desventajas dependiendo de cómo gestionan los procesos ligados al CPU y a los procesos de E/S.

1. El Round Robin tiene a ser menos eficiente en entornos mixtos de procesos ligados a CPU y E/S debido a su enfoque uniforme y la sobrecarga por cambios de contexto.
2. El SRTF aunque es más eficiente para minimizar el tiempo promedio de espera y respuesta, puede causar problemas significativos de hambruna y una alta sobrecarga en sistemas con alta variabilidad de procesos

13) Es posible que el quantum de un proceso nunca llegue a 0 en un esquema VRR con un contador de quantum variable. Esto puede suceder si el proceso cede la CPU voluntariamente, es preemptado antes de agotar su quantum, o si el quantum es reajustado dinámicamente. Aunque, en sistemas bien diseñados, este tipo de situaciones no afectan al funcionamiento general del planificador, ya que el objetivo principal es garantizar equidad y eficiencia en la ejecución de los procesos.

15) a) Para procesos interactivos, utilizaría Round Robin ya que estos procesos suelen requerir una respuesta rápida. El Round Robin garantiza una distribución equitativa del tiempo de CPU, con tiempo de respuesta cortos.

Para los procesos Batch, utilizaría FCFS o SJN (dependiendo si se tiene información previa sobre la duración de procesos) ya que estos no requieren una respuesta inmediata, aquí es más importante la eficiencia que la rapidez de respuesta.

b) En este caso utilizaría un algoritmo por Prioridades con Aging.

19)