ISO P4

▼ 1.

a) Un programa es una secuencia de instrucciones que existe desde que se edita hasta que se borra. Es estático. No tiene PC

Un proceso/job/tarea es un programa en ejecución. Puede estar ligado a la CPU (CPU Bound) o a E/S (I/O Bound). Existe desde que se lo ejecuta hasta que finaliza. Es dinámico. Tiene PC. Tiene PCB, que se crea cuando el padre del proceso hace fork (primero que se crea) y es lo último que se borra cuando el proceso finaliza

b) El tiempo de retorno (TR) es lo que tarda un proceso en irse en contraposicion al momento en que llega.

El tiempo de espera (TE) es el tiempo que reside un proceso en la cola (Ready queue)

c) El tiempo promedio de retorno (TPR) es el promedio de tiempo de retorno de un lote de procesos.

El tiempo promedio de espera (TPE) es el promedio de tiempo de espera de un lote de procesos

d) Un Quantum es una medida de tiempo que determina tiempos.

e) Un algoritmo apropiativo continua con la ejecución de un proceso hasta que éste termine. Un proceso monopoliza al procesador

Un algoritmo no apropiativo puede interrumpir la ejecución de un proceso para iniciar la ejecución de otro. Un proceso no monopoliza al procesador

f) Son tipos de planificadores

Short Term Scheduler determina que proceso pasará a ejecutarse a continuación. Hace que los procesos “compitan” por la CPU

Medium Term Scheduler realiza el swap/intecambio entre el disco y la memodia cuando el SO lo determina

Long Term Scheduler admine/agrega nuevos procesos a memoria

g) El dispatcher equilibra el tráfico de procesos a través de la carga y descarga

▼ 2.

a) I. top: muestra los procesos en ejecución a tiempo real (table of process)

II. htop: muestra la información crucial de los procesos en ejecución del sistema

III. ps: muestra los procesos en ejecución

IV. pstree: muestra los procesos en ejecución en modo de árbol.

V. kill: matar un proceso

VI. pggrepkillkillall: matar un proceso a traves de su nombre

VII. killall: matar un proceso a traves de su ID

VIII. renice: altera la prioridad de un proceso en ejecución

IX. xkill: matar un proceso en un servidor pasasndole el nombre o PID del proceso

X. atop: monitorear los recursos del sistema

b)

Las tres primeras lineas “#include” incluyen al archivo referenciado en el archivo actual. Son declaraciones externas.

Es un procedimiento llamado “main” (funcion principal) que no recibe parámetros (pero podría recibir) (no se que hace el int, capaz que determina que se devuelve un valor entero?). La funcion main() es la primera funcion que se crea cuando un programa es ejecutado. Crea una variable c entera, una variable pid_t con un pid. Imprime “Comienzo.: “ y en el siguiente renglon imprimirá proceso cuando se halla iterado 3 veces, y a cada iteración el pid toma el valor de la funcion fork() que crea un proceso. Luego devuelve 0.

I. -

II. - (no me funciona en bash)

c)

I. -

II. -

III. -

d)

I. Los pipes se utilizan para comunicar procesos. Puede utilizarse para comunicar al hijo y al padre. Un proceso hijo hereda los descriptores de ficheros abierto de su padre. De esta manera, el padre y el hijo comparten datos mediante el pipe. Tambien puede implementarse una comunicación bidireccional con un pipe para cada direccion (dos comunicaciones unidireccionales en sentido opuesto).

II. En C un pipe se crea con el system call “pipe()”. Una tuberia es un array con dos descriptores de fichero, que son enteros, uno para el extremo de escritura y otro para el extremo de lectura. Por lo tanto un pipe es un array de dos enteros. la funcion pipe() abre los dos descriptores y almacena su valor en los dos enteros. El primero es de lectura y el segundo de escritura, así funciona únicamente de manera unidireccional.

III. Para crear un pipe se requiere una arreglo de dos elementos.

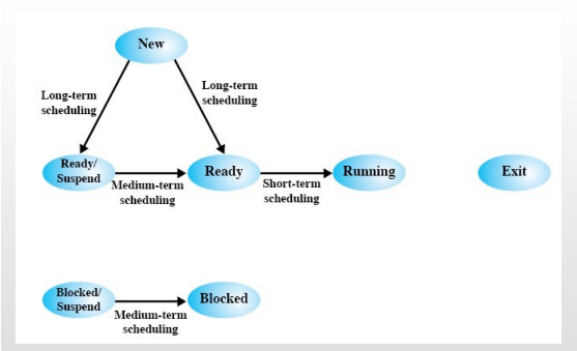
IV. es posible comunicar dos procesos de manera unidireccional.

e) Debe tener la información del contexto de un proceso (registros, PC, prioridad, etc). Esta info se almacena en la PCB

f) CPU Bound significa que la velocidad a la que avanza un proceso es limitada por la velocidad de la CPU. I/O Bound significa que la velocidad a la que avanza un proceso es limitada por la velocidad del subsistema de E

g) Los estados son: new (creación), ready (listo para ejecutarse), running (en ejecución) , waiting (espera a que se termine la interrupción para continuar su ejecución), terminated (finaliza. Se borra su PCB)

h/i)



▼ 3

a) (ni ganas de explicar con ejemplos)

FCFS: tambien llamado FIFO. Consiste en ejecutar los procesos en orden de llegada, similar a una cola. Siempre ejecuta el proceso que primero llega, o sea, el más longevo en la cola.

No favorece ningún tipo de proceso

Es de planificación no apropiativo

SJF: ejecuta el proceso cuyo tiempo de ejecución o ráfaga es más corta. Se conoce el tiempo que tarda por ejecución previa. Los procesos cortos se ejecutan antes que los largos

Los procesos largo pueden sufrir starvation/inanición debido a que, ante la aparición de muchos procesos cortos, pueden permanecer en cola mucho tiempo.

Es de planificación no apropiativa

Round Robin: consiste en una cola circular con tiempos establecidos para la ejecución de los procesos. Se toma el primer elemento y se ejecuta durante el tiempo que haya sido establecido. Este tiempo se mide en Quantums (Q). Una vez se llega al limite de Q establecido (a traves de un contador), se devuelve al proceso a la cola para tomar al siguiente. De esta manera ningun proceso monopoliza la CPU.

Es de planificación apropiativa.

Si el Q es pequeño, podremos llegar a una situación de “overhead”, en la que estaremos más tiempo haciendo context switch que ejecutando procesos. Si el Q es más grande que el tiempo que tardan los procesos en ejecutarse funciona igual que un FIFO

Prioridades: se selecciona el proceso de mayor prioridad para ejecutar. Cada proceso tiene una valor que indica su prioridad (a menor valor, mayor prioridad). Hay una cola para cada prioridad. Los procesos de baja prioridad pueden sufrir inanición, que puede solucionarse cambiandole la prioridad a un proceso o cambiando la cola de la que se toma el siguiente proceso.

Puede ser apropiativo o no apropiativo

b) Ninguno requiere de parámetros excepto Round Robin que necesita la definición de un Q.

c) FCFS no presenta ventaja alguna para un tipo de proceso en especifico.

SJF beneficia los I/O Bound

RR …

Prioridades no presenta ventajas para un tipo de proceso en especifico

d) RR presenta tiempos de retorno y de espera más altos que el resto

SJF requiere que se sepa la duración de su ráfaga de CPU, lo cual no siempre es asequible

▼ 4

a) Ambas variante del algoritmo de Round Robin requieren de una medida de tiempo (Quantum) que determine el tiempo que un algoritmo puede ejecutarse antes de volver a la cola. Las diferencias se presentan a la hora de determinar que sucede si un proceso finaliza previo a que el timer se resetee (O sea, llegue Q a 0). La variante

de timer fijo carga el siguiente proceso de la cola, que se ejecutará la cantidad de veces que faltó ejecutar el proceso anterior para que Q llegara a 0. La variante de timer variable reinicia el timer si un proceso termina antes que Q llegue a 0.

b)

Ejemplo RR Tiempo Fijo con Q=3

JOB	U de CPU	Llegada	0	1	2	3	4	5	6	7
1	5	0	>1	2	3					
2	6	1		>		1	2	3		
3	4	2			>				1	2
		Cola:	1	2	3	1	2	3	2	

Ejemplo RR Tiempo Variable con Q=3

JOB	U de CPU	Llegada	0	1	2	3	4	5	6	7
1	5	0	>1	2	3					
2	6	1		>		1	2	3		
3	4	2			>				1	2
		Cola:	1	2	3	1	2	3		

c) En RR fijo se puede marcar cada vez que se resetea Q

▼ 5

JOB	U de CPU	Llegada	0	1	2	3	4	5	6	7
1	7	0	>1	2	3	4	5	6	7<	
2	15	0	>							1
3	12	0	>							
4	4	0	>							
5	9	0	>							
FCFS		COLA	1	2	3	4	5			

JOB	U de CPU	Llegada	0	1	2	3	4	5	6	7
1	7	0	>				1	2	3	4
2	15	0	>							
3	12	0	>							
4	4	0	>1	2	3	4<				
5	9	0	>							
SJF		COLA	1	2	3	4	5			

JOB	U de CPU	Llegada	0	1	2	3	4	5	6	7
1	7	0	>1	2	3	4				
2	15	0	>				1	2	3	4
3	12	0	>							
4	4	0	>							
5	9	0	>							
RR TF = 4		COLA	1	2	3	4	5	1	2	3

Nota: cuando el JOB 2 quedó solo y termino el Q en 43, se encoló y tomó nuevamente para ejecutarlo

JOB	U de CPU	Llegada	0	1	2	3	4	5	6	7
1	7	0	>1	2	3	4				
2	15	0	>				1	2	3	4
3	12	0	>							
4	4	0	>							
5	9	0	>							
RR TV = 4		COLA	1	2	3	4	5	1	2	3

c) Round Robin de tiempo variable dió los tiempos más altos, casi igualados con Round Robin de tiempo fijo. SJF fue el que dió los tiempos más bajos seguido por FIFO

▼ 6

a/b) Ejercicio hecho en Excel

c) Los algoritmos de Round Robin siguen siendo los que peores prestaciones ofrecen. SJF es el mejor

d) El Quantum = 1 obliga a hacer muchos cambios entre el proceso que se esta ejecutando y la cola. Un Q más pequeño que las unidades de los procesos aumenta el tiempo de retorno y espera. Un Q más grande que los procesos mejora estos tiempos.

e) Usaría un valor de Q alto cuando los tiempos de CPU de los procesos son altos. De esta manera evitaría que se realicen muchos cambios de contexto entre ready y running y a su vez evitaría que un proceso monopolice la CPU

▼ 7

a) Diagrama realizado en Excel

b) SRTF redujo en gran medida el tiempo de espera y tiempo de retorno

▼ 8

El algoritmo no apropiativo (el proceso que toma lo ejecuta hasta terminarlo) mejora mucho los tiempos de ejecución y retorno. Lo utilizaría cuando las prioridades de los procesos son relevantes, por ejemplo con rutinas del sistema operativo. No es necesario implementar este algoritmo cuando las prioridades de los procesos son irrelevantes.

▼ 9

a) Inanición es morir de hambre. En términos de informática (y más precisamente hablando de procesos) hace referencia a cuando un proceso reside en la ready queue durante mucho tiempo.

b) La inanición puede ser provocada por los algoritmos SJF, SRTF y Prioridades, debido a que son algoritmos que dan prioridad a la ejecución de procesos cuyo valor de un atributo es más extremo (tiempo de CPU más bajo en SJF, tiempo restante más bajo en SRTF, y nivel de prioridad más alto en Prioridades)

c) La inanición puede solucionarse estableciendo algoritmos que hagan excepciones. En prioridades puede cambiarse la prioridad de un proceso o tomar procesos de otra cola de nivel de prioridad pasado determinado tiempo.

Otros (que se me ocurren) pueden ser que pasado un tiempo en la cola sea llamado a ejecutarse, o que el proceso se divida en procesos más pequeños.

▼ 10

Ejemplo Job 4 (R3, 1, 2): luego de su instante de CPU 1, el job4 se va a R3 durante 2 instantes. O sea quiere ejecutar 1 unidad de CPU, y luego irse al recurso R3 durante 2 instantes.

Notas/Observaciones:

- En FCFS cuando un proceso toma un recurso, se ejecuta sobre ese recurso y mientras lo está haciendo el siguiente proceso en la cola arranca a ejecutarse.
- No pueden haber dos procesos tomando el mismo recurso a la vez.
- El ejemplo 2 de qplanif parece ser:

JOB	LLEGADA	CPU	I/O
1	0	5	(R1, 3, 2)
2	1	4	(R2, 5, 2)
3	2	3	(R3, 7, 3)

  Aunque no termino de comprender si la toma de recursos cuenta como tiempo de CPU (O sea, seria CPU (JOB 1)= 7, CPU(JOB 2) = 6 y CPU (JOB 3) = 5) o es externo

- Si dos procesos se encolan a la vez, si uno se re-encola y otro esta llegando, se pone primero el de primera llegada, o sea el más longevo. Si dos llegan a la vez, se encolan por PID. Son los dos criterios de “desempate”, orden de llegada o longevidad y PID
- Si un I/O se ejecuta en el instante 1, se hace un tiempo de CPU y despues se toma el recurso o no se hace ningun tiempo de CPU?
  - RTA: el I/O se ejecuta LUEGO DEL INSTANTE 1, es decir, el que sería el instante 2 de CPU se utiliza para tomar el recurso.

▼ 11

En Round Robin, cuando las operaciones I/O se dan antes de que se resetee el Quantum, provoca que los procesos no lleguen a completar los tiempos de Q y se den más context switchs de los que debería. Si hay muchas operaciones I/O, da igual que tan grande sea Q que van a haber muchos context switch

En SRTF, si hay muchas operaciones I/O, se analizará mucho más seguido cual es el proceso cuyo tiempo de espera es menor, lo que provocará que posiblemente se alterne entre procesos muy seguido (muchos context switch), haciendo que un proceso corto tarde mucho más en finalizarse (aumenta el TE) *no estoy seguro de esto último*.

▼ 13

Puede suceder si el tiempo de CPU de un proceso es menor al Quantum inicial

▼ 14

ni idea

▼ 15

a) Para procesos interactivos utilizaría un algoritmo que ejecute y devuelva los procesos lo más rápido posible, entonces eligiría SJF o SRTF, debido a que son los que mejores tiempos de espera y retorno ofrecen. (Podría ser RR?)

Para los procesos batch (procesamiento en lotes) quiero elegir un algoritmo que, sin importar el TE y TR, termine con el lote o bloque lo antes posible, sin interes en alternar, dar prioridades o darle su lugar a cada proceso. Usaría FIFO/FSFC

b) Utilizaría prioridades implementando Aging, para evitar la inanición de procesos.

▼ 18

Implementaría un algoritmo tipo Round Robin (da igual si tiempo fijo o variable) con un Quantum bajo, por ejemplo 4, combinado con colas de prioridad. La manera de funcionar sería que si un algoritmo completa el ciclo de Quantum (es decir Quantum llega a 0) y no terminó, se envía a una cola de menor prioridad. También agregaría un

contador de veces que el algoritmo fue enviado a la cola, así si llega determinado número se envía a una cola de aún menor prioridad. Para que no sufra inanición, se le cambiará de cola de prioridad a una mayor cuando hayan pasado X unidades de tiempo, por ejemplo 4, sin llamarse a ejecución

nota: lo pensé en base a que no se sabe cuánto dura el proceso en ejecutarse.

▼ 19

que se yo amigo re difici

▼ 20

ni idea amigo re rebuscado tengo cosas mas importantes que hacer

▼ 21

En RR, si el Q es alto se aproxima a FIFO. Esto es porque es posible que si el Q es alto, los procesos se realicen en su totalidad dentro del ciclo, entonces el algoritmo pasaría a ejecutar los procesos completos según provienen de la cola, igual a …

▼ 22

a) Típicamente los procesadores de las máquinas actuales son homogéneos, aunque cada vez hay más heterogéneos (segun wikipedia). Son fuertemente acoplados, porque comparten RAM y SO.

b) Significa que no hay preferencia de procesador al que asignarle un proceso, se dirigirá los procesos de manera uniforme, balanceada y coordinada.

c) Significa que un dispositivo actúa como maestro, el cual controla a otros dispositivos esclavos y sirve como centro de comunicación.

▼ 23

a) La manera más sencilla sería a través de la asignación dinámica, la cual balancea la carga entre los procesadores, no existen preferencias.

b) (no chequeado) Las ventajas pueden ser que el algoritmo es más sencillo de implementar y no hace falta hacer filtros ni chequear información ya sea del proceso o del procesador para saber a que procesador va un proceso.

(Esta respuesta no está bien porque los procesadores homogéneos no tienen diferencias, entonces que prefiera a uno u otro no es por eficiencia) Las desventajas pueden ser que no se está aprovechando al máximo las capacidades del CPU en relación a las características del proceso, con el objetivo de hacer el procesamiento lo más eficiente posible. Probablemente si un proceso tiene más afinidad con un procesador específico, su procesamiento sería más rápido/usaría menos recursos, pero como no se tiene a la afinidad en consideración, no se llega al mejor nivel de eficiencia.

Las desventajas pueden ser peor organización. (Seguramente no lo es)

▼ 24

a) La huella es la información que deja un proceso en la cache del procesador

b) La afinidad con un procesador por parte de un proceso consiste en la preferencia por uno u otro para ser procesado

c) El hecho de que exista información sobre el proceso en la cache del procesador puede facilitar el procesamiento.

d) Windows: se puede cambiar en el administrador de tareas

Linux/GNU: se puede cambiar o asignar con el comando taskset

e) ne

f) Son conceptos contrapuestos. Mientras que el balanceo busca la igual distribución de procesos, la afinidad determina que un proceso se corresponde con un procesador. El balanceo provoca que todos los procesadores esten igualmente cargados. La afinidad provoca que halla desbalance y que suceda que un procesador esté sobrecargado y otro sin carga. Si cambias procesos por ingresos estas diferencias podrían iniciar una Guerra Fría entre arquitecturas de computadoras.

▼ 25

La más performante es la implementada en B