UNIVERSIDAD NACIONAL DE LA PLATA

Introducción a los Sistemas Operativos

Práctica 4

Objetivo

El objetivo de esta práctica es que el alumno comprenda los aspectos base acerca de la planificación de procesos en un Sistema Operativo (tipos de planificadores, algoritmos y sus variantes, etc.). Además, para la autocorrección de los ejercicios, es deseable la utilización del simulador[1] alojado en el repositorio *GitHub* de la cátedra.

1. Comandos para manejo de procesos
   (a) Investigue y detalle para que sirve cada uno de los siguientes comandos. (Puede pasar que algún comando no venga por defecto en su distribución por lo que deberá instalarlo):
      I. *top*
      II. *htop*
      III. *ps*
      IV. *pstree*
      V. *kill*
      VI. *pgrep*
      VII. *pkill*
      VIII. *killall*
      IX. *renice*
      X. *xkill*
      XI. *atop*

2. Creación de procesos - Fork
   (a) Observe detenidamente el siguiente código. Intente entender lo que hace sin necesidad de ejecutarlo.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(void) {
int c;
pid_t pid;
printf("Comienzo.:\n");
for (c = 0; c < 3 ; c++ )
{
        pid = fork();
}
printf("Proceso\n");
return 0;
}
```

[1] https://github.com/unlp-so/contenidos/blob/master/practicas/iso/practica4/Simulador.tar.gz

i. *¿Cuántas líneas con la palabra "Proceso" aparecen al final de la ejecución de este programa?*
ii. *¿El número de líneas es el número de procesos que han estado en ejecución?*
iii. *Ejecute el programa y compruebe si su respuesta es correcta. Modifique el valor del bucle for y compruebe los nuevos resultados*

(b) Vamos a tomar una variante del programa anterior. Ahora, además de un mensaje, vamos a añadir una variable y, al final del programa vamos a mostrar su valor. El nuevo código del programa se muestra a continuación.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(void) {
int c;
int p=0;
pid_t pid;
for (c = 0; c < 3 ; c++ )
{
        pid = fork();
}
p++;
printf("Proceso %d\n", p);
return 0;
}
```

i. *¿Qué valores se muestran por consola?*
ii. *¿Cuál es el valor (o valores) que aparece? Analice y explique.*
iii. *Ejecute el programa y compruebe si su respuesta es correcta. Modifique el valor del bucle for y el lugar dónde se incrementa la variable p y compruebe los nuevos resultados*

3. Comunicación entre procesos
   (a) Investigue la forma de comunicación entre procesos a través de pipes
   (b) ¿Que parametro es necesario para la creación de un pipe?. Explique para que se utiliza
   (c) ¿Que tipo de comunicación es posible con pipes?

4. Responda en forma sintética sobre los siguientes conceptos:
   (a) Programa y Proceso.
   (b) Defina Tiempo de retorno (**TR**) y Tiempo de espera (**TE**) para un Job.
   (c) Defina Tiempo Promedio de Retorno (**TPR**) y Tiempo promedio de espera (**TPE**) para un lote de JOBS.
   (d) ¿Qué es el Quantum?
   (e) ¿Qué significa que un algoritmo de scheduling sea apropiativo o no apropiativo (Preemptive o Non-Preemptive)?
   (f) ¿Qué tareas realizan?:
      i. *Short Term Scheduler*
      ii. *Long Term Scheduler*
      iii. *Medium Term Scheduler*

(g) ¿Qué tareas realiza el *Dispatcher*?

5. Procesos:
   (a) ¿Cuál es la información mínima que el SO debe tener sobre un proceso?¿En que estructura de datos asociada almacena dicha información?
   (b) ¿Qué significa que un proceso sea "CPU Bound" y "I/O Bound"?
   (c) ¿Cuáles son los estados posibles por los que puede atravesar un proceso?
   (d) Explique mediante un diagrama las posibles transiciones entre los estados.
   (e) ¿Que scheduler de los mencionados en 4 f se encarga de las transiciones?

6. Para los siguientes algoritmos de scheduling:
   - *FCFS* (Fisrt Coome First Served)
   - *SJF* (Shortest Job First)
   - *Round Robin*
   - *Prioridades*

   (a) Explique su funcionamiento mediante un ejemplo.
   (b) ¿Alguno de ellos requiere algún parámetro para su funcionamiento?
   (c) Cual es el mas adecuado según los tipos de procesos y/o SO.
   (d) Cite ventajas y desventajas de su uso.

7. Para el algoritmo Round Robin, existen 2 variantes:
   - **Timer Fijo**
   - **Timer Variable**

   (a) ¿Qué significan estas 2 variantes?
   (b) Explique mediante un ejemplo sus diferencias.
   (c) En cada variante ¿Dónde debería residir la información del Quantum?

8. Se tiene el siguiente lote de procesos que arriban al sistema en el instante 0 (cero):

Job	Unidades de CPU
1	7
2	15
3	12
4	4
5	9

   (a) Realice los diagramas de Gantt según los siguientes algoritmos de scheduling:
      i. FCFS (First Come, First Served)
      ii. SJF (Shortest Job First)
      iii. Round Robin con quantum = 4 y Timer Fijo
      iv. Round Robin con quantum = 4 y Timer Variable
   (b) Para cada algoritmo calcule el TR y TE para cada job así como el TPR y el TPE.
   (c) En base a los tiempos calculados compare los diferentes algoritmos.

9. Se tiene el siguiente lote de procesos:

Job	Llegada	Unidades de CPU
1	0	4
2	2	6
3	3	4
4	6	5
5	8	2

(a) Realice los diagramas de Gantt según los siguientes algoritmos de scheduling:
   i. FCFS (First Come, First Served)
   ii. SJF (Shortest Job First)
   iii. Round Robin con quantum = 1 y Timer Variable
   iv. Round Robin con quantum = 6 y Timer Variable

(b) Para cada algoritmo calcule el TR y TE para cada job así como el TPR y el TPE.

(c) En base a los tiempos calculados compare los diferentes algoritmos.

(d) En el algoritmo Round Robin, que conclusión se puede sacar con respecto al valor del quantum.

(e) ¿Para el algoritmo Round Robin, en que casos utilizaría un valor de quantum alto y que ventajas y desventajas obtendría?

10. Una variante al algoritmo SJF es el algoritmo SJF apropiativo o SRTF (Shortest Remaining Time First):

(a) Realice el diagrama del Gantt para este algoritmo según el lote de trabajos del ejercicio 9.

(b) ¿Nota alguna ventaja frente a otros algoritmos?

11. Suponga que se agregan las siguientes prioridades al lote de procesos del ejercicio 9, donde un menor número indica mayor prioridad:

Job	Prioridad
1	3
2	4
3	2
4	1
5	2

(a) Realice el diagrama de Gantt correspondiente al algoritmo de planificación por prioridades según las variantes:
   i. No Apropiativa
   ii. Apropiativa

(b) Calcule el TR y TE para cada job así como el TPR y el TPE.

(c) ¿Nota alguna ventaja frente a otros algoritmos? Bajo que circunstancias lo utilizaría y ante que situaciones considera que la implementación de prioridades podría no ser de mayor relevancia?

12. Inanición (*Starvation*)

(a) ¿Qué significa?

(b) ¿Cuál/es de los algoritmos vistos puede provocarla?

(c) ¿Existe alguna técnica que evite la inanición para el/los algoritmos mencionados en b?

13. Los procesos, durante su ciclo de vida, pueden realizar operaciones de I/O como lecturas o escrituras a disco, cintas, uso de impresoras, etc.

El SO mantiene para cada dispositivo, que se tiene en el equipo, una cola de procesos que espera por la utilización del mismo (al igual que ocurre con la Cola de Listos y la CPU, ya que la CPU es un dispositivo mas).

Cuando un proceso en ejecución realiza una operación de I/O el mismo es expulsado de la CPU y colocado en la cola correspondiente a el dispositivo involucrado en la operación.

El SO dispone también de un "I/O Scheduling" que administrada cada cola de dispositivo a través de algún algoritmo (FCFS, Prioridades, etc.). Si al colocarse un proceso en la cola del dispositivo, la misma se encuentra vacía el mismo será atendido de manera inmediata, caso contrario, deberá esperar a que el SO lo seleccione según el algoritmo de scheduling establecido.

Los mecanismos de I/O utilizados hoy en día permiten que la CPU no sea utilizada durante la operación, por lo que el SO puede ejecutar otro proceso que se encuentre en espera una vez que el proceso bloqueado por la I/O se coloca en la cola correspondiente.

Cuando el proceso finaliza la operación de I/O el mismo retorna a la cola de listos para competir nuevamente por la utilización de la CPU.

Para los siguientes algoritmos de Scheduling:

- FCFS
- Round Robin con quantum = 2 y timer variable.

Y suponiendo que la cola de listos de todos los dispositivos se administra mediante FCFS, realice los diagramas de Gantt según las siguientes situaciones:

(a) Suponga que al lote de procesos del ejercicio 9 se agregan las siguientes operaciones de entrada salida:

Job	**I/O (rec,ins,dur)**
1	(R1, 2, 1)
2	(R2, 3, 1) (R2, 5, 2)
4	(R3, 1, 2) (R3, 3, 1)

(b) Suponga que al lote de procesos del ejercicio 9 se agregan las siguientes operaciones de entrada salida:

Job	**I/O (rec,ins,dur)**
1	(R1, 2, 3) (R1, 3, 2)
2	(R2, 3, 2)
3	(R2, 2, 3)
4	(R1, 1, 2)

14. Algunos algoritmos pueden presentar ciertas desventajas cuando en el sistema se cuenta con procesos ligados a CPU y procesos ligados a entrada salida. Analice las mismas para los siguientes algoritmos:

(a) Round Robin

(b) SRTF (Shortest Remaining Time First)

15. Para equiparar la desventaja planteada en el ejercicio 14), se plantea la siguiente modificación al algoritmo:

    Algoritmo VRR (Virtual Round Robin): Este algoritmo funciona igual que el Round Robin, con la diferencia que cuando un proceso regresa de una I/O se coloca en una cola auxiliar. Cuando se tiene que tomar el próximo proceso a ejecutar, los procesos que se encuentra en la cola auxiliar tienen prioridad sobre los otros. Cuando se elije un proceso de la cola auxiliar se le otorga el procesador por tantas unidades de tiempo como le falto ejecutar en su ráfaga de CPU anterior, esto es, se le otorga la CPU por un tiempo que surge entre la diferencia del quantum original y el tiempo usado en la última ráfaga de CPU.

    (a) Analice el funcionamiento de este algoritmo mediante un ejemplo. Marque en cada instante en que cola se encuentran los procesos.

    (b) Realice el ejercicio 10)a) nuevamente considerando este algoritmo, con un quantum de 2 unidades y Timer Variable.

16. Suponga que un SO utiliza un algoritmo de VRR con Timer Variable para el planificar sus procesos. Para ello, el quantum es representado por un contador, que es decrementado en 1 unidad cada vez que ocurre una interrupción de reloj. ¿Bajo este esquema, puede suceder que el quantum de un proceso nunca llegue a 0 (cero)? Justifique su respuesta.

17. El algoritmo SJF (y SRTF) tiene como problema su implementación, dada la dificultad de conocer la duración de la próxima ráfaga de CPU. Es posible realizar una estimación de la próxima, utilizando la media de las ráfagas de CPU para cada proceso.

    Así, por ejemplo, podemos tener la siguiente formula:

    $$S_{n+1} = \frac{1}{n} T_n + \frac{n-1}{n} S_n \tag{1}$$

    Donde:

    T_i = duración de la ráfaga de CPU i-ésima del proceso.

    S_i = valor estimado para el i-ésimo caso

    S_i = valor estimado para la primer ráfaga de CPU. No es calculado.

    (a) Suponga un proceso cuyas ráfagas de CPU reales tienen como duración: 6, 4, 6, 4, 13, 13, 13 Calcule que valores se obtendrían como estimación para las ráfagas de CPU del proceso si se utiliza la formula 1, con un valor inicial estimado de S_1=10.
    La formula anterior 1 le da el mismo peso a todos los casos (siempre calcula la media). Es posible reescribir la formula permitiendo darle un peso mayor a los casos mas recientes y menor a casos viejos (o viceversa). Se plantea la siguiente formula:

    $$S_{n+1} = \alpha T_n + (1-\alpha) S_n \tag{2}$$

    Con $0 \langle \alpha \langle 1$.

    (b) Analice para que valores de α se tienen en cuenta los casos mas recientes.

    (c) Para la situación planteada en a) calcule que valores se obtendrían si se utiliza la formula 2 con $\alpha = 0,2$; $\alpha = 0,5$ y $\alpha = 0,8$.

    (d) Para todas las estimaciones realizadas en a y c ¿Cuál es la que mas se asemeja a las ráfagas de CPU reales del proceso?

18. Colas Multinivel

    Hoy en día los algoritmos de planificación vistos se han ido combinando para formar algoritmos más eficientes. Así surge el algoritmo de Colas Multinivel, donde la cola de procesos listos es divida en varias colas, teniendo cada una su propio algoritmo de planificación.

(a) Suponga que se tienen dos tipos de procesos: *Interactivos* y *Batch*. Cada uno de estos procesos se coloca en una cola según su tipo. ¿Qué algoritmo de los vistos utilizaría para administrar cada una de estas colas?.

A su vez, se utiliza un algoritmo para administrar cada cola que se crea. Así, por ejemplo, el algoritmo podría determinar mediante prioridades sobre que cola elegir un proceso.

(b) Para el caso de las dos colas vistas en a: ¿Qué algoritmo utilizaría para planificarlas?

19. Suponga que en un SO se utiliza un algoritmo de planificación de colas multinivel. El mismo cuenta con 3 colas de procesos listos, en las que los procesos se encolan en una u otra según su prioridad. Hay 3 prioridades (1 , 2 , 3), donde un menor número indica mayor prioridad.
Se utiliza el algoritmo de prioridades para la administración entre las colas.

Se tiene el siguiente lote de procesos a ser procesados con sus respectivas operaciones de I/O:

Job	Llegada	CPU	I/O (rec,ins,dur)	Prioridad
1	0	9	(R1, 4, 2) (R2, 6, 3) (R1, 8, 3)	1
2	1	5	(R3, 3, 2) (R3, 4, 2)	2
3	2	5	(R1, 4, 1)	3
4	3	7	(R2, 1, 2) (R2, 5, 3)	2
5	5	5	(R1, 2, 3) (R3, 4, 3)	1

Suponiendo que las colas de cada dispositivo se administran a trabes de FCFS y que cada cola de procesos listos se administra por medio de un algoritmo RR con un quantum de 3 unidades y Timer Variable, realice un diagrama de Gantt:

(a) Asumiendo que NO hay apropiación entre los procesos.

(b) Asumiendo que hay apropiación entre los procesos.

20. En el esquema de Colas Multinivel, cuando se utiliza un algoritmo de prioridades para administrar las diferentes colas los procesos pueden sufrir starvation.

La técnica de envejecimiento se puede aplicar a este esquema, haciendo que un proceso cambie de una cola de menor prioridad a una de mayor prioridad, después de cierto periodo de tiempo que el mismo se encuentra esperando en su cola. Luego de llegar a una cola en la que el proceso llega a ser atendido, el mismo retorna a su cola original.

Por ejemplo: Un proceso con prioridad 3 esta en cola su cola correspondiente. Luego de X unidades de tiempo, el proceso se mueve a la cola de prioridad 2. Si en esta cola es atendido, retorna a su cola original, en caso contrario luego de sucederse otras X unidades de tiempo el proceso de mueve a la cola de prioridad 1. Esta última acción se repite hasta que el proceso obtiene la CPU, situación que hace que el mismo vuelva a su cola original.

(a) Para los casos a y b del ejercicio 19 realice el diagrama de Gantt considerando además que se tiene un envejecimiento de 4 unidades.

21. La situación planteada en el ejercicio 20, donde un proceso puede cambiar de una cola a otra, se la conoce como Colas Multinivel con Realimentación.

Suponga que se quiere implementar un algoritmo de planificación que tenga en cuenta el tiempo de ejecución consumido por el proceso, penalizando a los que más tiempo de ejecución tienen. (Similar a la tarea del algoritmo SJF que tiene en cuenta el tiempo de ejecución que resta).

Utilizando los conceptos vistos de Colas Multinivel con Realimentación indique que colas implementaría, que algoritmo usaría para cada una de ellas así como para la administración de las colas entre sí.

Tenga en cuenta que los procesos no deben sufrir inanición.

22. Un caso real: "Unix Clasico " (SVR3 y BSD 4.3)

Estos sistemas estaban dirigidos principalmente a entornos interactivos de tiempo compartido. El algoritmo de planificación estaba diseñado para ofrecer buen tiempo de respuesta a usuarios interactivos y asegurar que los trabajos de menor prioridad (en segundo plano) no sufrieran inanición.

La planificación tradicional usaba el concepto de colas multinivel con realimentación, utilizando RR para cada uno de las colas y realizando el cambio de proceso cada un segundo (quantum). La prioridad de cada proceso se calcula en función de la clase de proceso y de su historial de ejecución. Para ello se aplican las siguientes funciones:

$$CPU_j(i) = \frac{CPU_j(i-1)}{2} \tag{3}$$

$$P_j(i) = Base_j + \frac{CPU_j(i)}{2} + nice_j \tag{4}$$

donde:

$CPU_j(i)$ = Media de la utilización de la CPU del proceso j en el intervalo i.

$P_j(i)$ = Prioridad del proceso j al principio del intervalo i (los valores inferiores indican prioridad más alta).

$Base_j$ = Prioridad base del proceso j.

$Nice_j$ = Factor de ajuste.

La prioridad del proceso se calcula cada segundo y se toma una nueva decisión de planificación. El propósito de la prioridad base es dividir los procesos en bandas fijas de prioridad. Los valores de CPU y nice están restringidos para impedir que un proceso salga de la banda que tiene asignada. Las bandas definidas, en orden decreciente de prioridad, son:

- Intercambio
- Control de Dispositivos de I/O por bloques
- Gestión de archivos
- Control de Dispositivos de I/O de caracteres
- Procesos de usuarios

Veamos un ejemplo: Supongamos 3 procesos creados en el mismo instante y con prioridad base 60 y un valor nice de 0. El reloj interrumpe al sistema 60 veces por segundo e incrementa un contador para el proceso en ejecución.

Los sectores en celeste representan el proceso en ejecución.

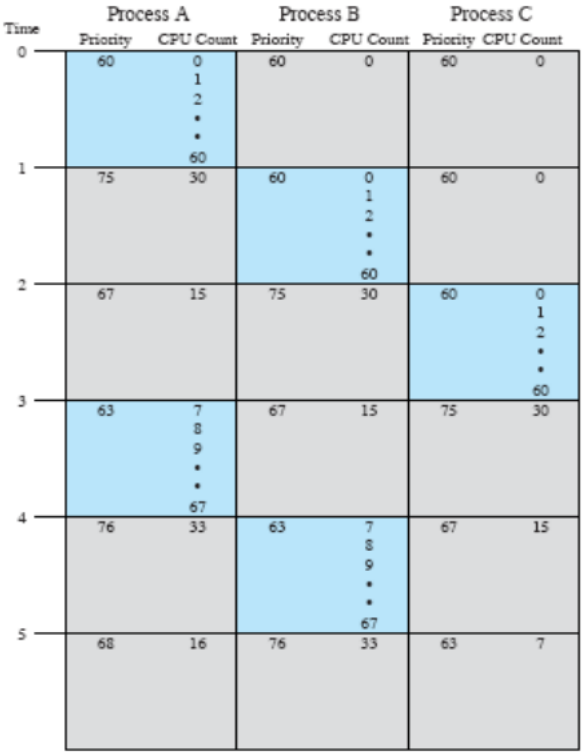
(a) Analizando la jerarquía descripta para las bandas de prioridades: ¿Que tipo de actividad considera que tendrá más prioridad? ¿Por qué piensa que el scheduler prioriza estas actividades?

(b) Para el caso de los procesos de usuarios, y analizando las funciones antes descriptas: ¿Qué tipo de procesos se encarga de penalizar? (o equivalentemente se favorecen). Justifique

(c) La utilización de RR dentro de cada cola: ¿Verdaderamente favorece al sistema de Tiempo Compartido? Justifique.

23. A cuáles de los siguientes tipos de trabajos:

(a) cortos acotados por CPU



Time	Process A Priority	Process A CPU Count	Process B Priority	Process B CPU Count	Process C Priority	Process C CPU Count
0	60	0 1 2 • • 60	60	0	60	0
1	75	30	60	0 1 2 • • 60	60	0
2	67	15	75	30	60	0 1 2 • • 60
3	63	7 8 9 • • 67	67	15	75	30
4	76	33	63	7 8 9 • • 67	67	15
5	68	16	76	33	63	7

(b) cortos acotados por E/S

(c) largos acotados por CPU

(d) largos acotados por E/S

benefician las siguientes estrategias de administración:

(a) prioridad determinada estáticamente con el método del más corto primero (SJF).

(b) prioridad dinámica inversamente proporcional al tiempo transcurrido desde la última operación de E/S.

24. Explicar porqué si el quantum "q.[en] Round-Robin se incrementa sin límite, el método se aproxima a FIFO.

25. Los sistemas multiprocesador pueden clasificarse en:

- **Homogéneos**: Los procesadores son iguales. Ningún procesador tiene ventaja física sobre el resto.
- **Heterogéneos**: Cada procesador tiene su propia cola y algoritmo de planificación.

Otra clasificación posible puede ser:

- **Multiprocesador débilmente acoplados**: Cada procesador tiene su propia memoria principal y canales.
- **Procesadores especializados**: Existe uno o más procesadores principales de propósito general y varios especializados controlados por el primero (ejemplo procesadores de E/S, procesadores Java, procesadores Criptográficos, etc.).
- **Multiprocesador fuertemente acoplado**: Consta de un conjunto de procesadores que comparten una memoria principal y se encuentran bajo el control de un Sistema Operativo

(a) ¿Con cuál/es de estas clasificaciones asocia a las PCs de escritorio habituales?

(b) ¿Qué significa que la asignación de procesos se realice de manera simétrica?

(c) ¿Qué significa que se trabaje bajo un esquema Maestro/esclavo?

26. Asumiendo el caso de procesadores homogéneos:

(a) ¿Cuál sería el método de planificación más sencillo para asignar CPUs a los procesos?

(b) Cite ventajas y desventajas del método escogido

27. Indique brevemente a que hacen referencia los siguientes conceptos:

(a) Huella de un proceso en un procesador

(b) Afinidad con un procesador

(c) ¿Por qué podría ser mejor en algunos casos que un proceso se ejecute en el mismo procesador?

(d) ¿Puede el usuario en *Windows* cambiar la afinidad de un proceso? ¿y en *GNU/Linux*?

(e) Investigue el concepto de balanceo de carga (load balancing).

(f) Compare los conceptos de afinidad y balanceo de carga y como uno afecta al otro.

28. Si a la tabla del ejercicio 9 la modificamos de la siguiente manera: Y considerando que el

Job	**Llegada**	**CPU**	**Afinidad**
1	0	4	CPU0
2	2	6	CPU0
3	3	4	CPU1
4	6	5	CPU1
5	8	2	CPU0

scheduler de los Sistemas Operativos de la familia *Windows* utiliza un mecanismo denominado preferred processor (procesador preferido). El scheduler usa el procesador preferido a modo de afinidad cuando el proceso esta en estado ready. De esta manera el sheduler asigna este procesador a la tarea si este está libre.

(a) Ejecute el esquema anterior utilizando el algoritmo anterior.

(b) Ejecute el esquema anterior. Pero ahora si el procesador preferido no está libre es asignado a otro procesador. Luego el procesador preferido de cada job es el último en el cual ejecuto.

(c) Para cada uno de los casos calcule el tiempo promedio de retorno y el tiempo promedio de espera.

(d) ¿Cuál de las dos alternativas planteadas es mas performante?