Conceptos de Sistemas Operativos (IC)

Introducción a los Sistemas Operativos (LI, LS, APU, ATIC)

Trabajo Práctico N° 1

Objetivo

Comprender aspectos básicos de los sistemas operativos, los procesos y su planificación para la administración de la CPU. El objetivo de esta práctica es que el alumno se familiarice con los conceptos básicos del sistema operativo *GNU/Linux*, su instalación, entorno y comandos principales.

Se recomienda, para la autocorrección de los ejercicios, la utilización del simulador de planificación que se publicará en la plataforma.

Temas Incluidos

Definición de SO. Componentes de un SO. Apoyo del Hardware: Modos de ejecución, interrupciones. Llamadas al sistema. Programa y Proceso. Planificación de CPU. Colas de planificación. Context Switch. Creación de procesos. Linux, instalación y conceptos básicos.

1. Características de *GNU/Linux*:
   a. Mencione y explique las características más relevantes de *GNU/Linux*.
   b. Mencione otros sistemas operativos y compárelos con *GNU/Linux* en cuanto a los puntos mencionados en el inciso *a*.
   c. ¿Qué es **GNU**?
   d. Indique una breve historia sobre la evolución del proyecto *GNU.*
   e. Explique qué es la multitarea, e indique si *GNU/Linux* hace uso de ella.
   f. ¿Qué es **POSIX**?

2. Distribuciones de *GNU/Linux*:
   a. ¿Qué es una distribución de *GNU/Linux*? Nombre al menos 4 distribuciones de *GNU/Linux* y cite diferencias básicas entre ellas.
   b. ¿En qué se diferencia una distribución de otra?
   c. ¿Qué es **Debian**? Acceda al sitio https://www.debian.org/ e indique cuáles son los objetivos del proyecto y una breve cronología del mismo.

3. Estructura de *GNU/Linux*:
   a. Nombre cuales son los 3 componentes fundamentales de *GNU/Linux*.
   b. Mencione y explique la estructura básica del Sistema Operativo *GNU/Linux*.

4. Kernel:

a. ¿Cuáles son sus funciones principales?
b. ¿Cuál es la versión actual? ¿Cómo se definía el esquema de versionado del Kernel en versiones anteriores a la 2.4? ¿Qué cambió en el versionado que se impuso a partir de la versión 2.6?
c. ¿Es posible tener más de un Kernel de *GNU/Linux* instalado en la misma máquina?
d. ¿Dónde se encuentra ubicado dentro del File System?

5. Intérprete de comandos *(Shell)*:
   a. ¿Qué es?
   b. ¿Cuáles son sus funciones?
   c. Mencione al menos 3 intérpretes de comandos que posee *GNU/Linux* y compárelos entre ellos.
   d. ¿Dónde se ubican (*path*) los comandos propios y externos al Shell?
   e. ¿Por qué considera que el Shell no es parte del Kernel de *GNU/Linux*?
   f. ¿Es posible definir un intérprete de comandos distinto para cada usuario? ¿Desde dónde se define? ¿Cualquier usuario puede realizar dicha tarea?

6. El sistema de Archivos *(File System)* en Linux:
   a. ¿Qué es?
   b. ¿Cuál es la estructura básica de los File System en *GNU/Linux*? Mencione los directorios más importantes e indique qué tipo de información se encuentra en ellos. ¿A qué hace referencia la sigla **FHS**?
   c. Mencione sistemas de archivos soportados por GNU/Linux.
   d. ¿Es posible visualizar particiones del tipo FAT y NTFS (que son de Windows) en GNU/Linux?

7. Particiones:
   a. Definición. Tipos de particiones. Ventajas y Desventajas.
   b. ¿Cómo se identifican las particiones en *GNU/Linux*? (Considere discos **IDE**, **SCSI** y **SATA**).
   c. ¿Cuántas particiones son necesarias como mínimo para instalar *GNU/Linux*? Nómbrelas indicando tipo de partición, identificación, tipo de File System y punto de montaje.
   d. Dar ejemplos de diversos casos de particionamiento dependiendo del tipo de tarea que se deba realizar en su sistema operativo.
   e. ¿Qué tipo de software para particionar existe? Menciónelos y compare.

8. Arranque (*bootstrap*) de un Sistema Operativo:
   a. ¿Qué es el **BIOS**? ¿Qué tarea realiza?
   b. ¿Qué es **UEFI**? ¿Cuál es su función?
   c. ¿Qué es el **MBR**? ¿Qué es el **MBC**?
   d. ¿A qué hacen referencia las siglas **GPT**? ¿Qué sustituye? Indique cuál es su

formato.

e. ¿Cuál es la funcionalidad de un "Gestor de Arranque"? ¿Qué tipos existen? ¿Dónde se instalan? Cite gestores de arranque conocidos.
f. ¿Cuáles son los pasos que se suceden desde que se prende una computadora hasta que el Sistema Operativo es cargado (proceso de *bootstrap*)?
g. Analice el proceso de arranque en *GNU/Linux* y describa sus principales pasos.
h. ¿Cuáles son los pasos que se suceden en el proceso de parada (*shutdown*) de *GNU/Linux*?
i. ¿Es posible tener en una PC *GNU/Linux* y otro Sistema Operativo instalado? Justifique.

9. Archivos y editores:
   a. ¿Cómo se identifican los archivos en *GNU/Linux*?
   b. Investigue el funcionamiento de los editores **vim, mano** y **mcedit**, y los comandos **cat, more y less**.
   c. Cree un archivo llamado "prueba.exe" en su directorio personal usando el **vim**. El mismo debe contener su número de alumno y su nombre.
   d. Investigue el funcionamiento del comando **file**. Pruébelo con diferentes archivos. ¿Qué diferencia nota?
   e. Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el uso de archivos:
      i. cd
      ii. mkdir
      iii. rmdir
      iv. ln
      v. df
      vi. tail
      vii. who
      viii. ls
      ix. pwd
      x. cp
      xi. mv
      xii. find

10. Indique qué comando es necesario utilizar para realizar cada una de las siguientes acciones. Investigue su funcionamiento y parámetros más importantes:
    a. Cree la carpeta **ISOCSO**
    b. Acceda a la carpeta
    c. Cree dos archivos con los nombres **isocso.txt** e **isocso.csv**
    d. Liste el contenido del directorio actual
    e. Visualizar la ruta donde estoy situado
    f. Busque todos los archivos en los que su nombre contiene la cadena "iso*"
    g. Informar la cantidad de espacio libre en disco
    h. Verifique los usuarios conectado al sistema
    i. Editar a el archivo **isocso.txt** e ingresar Nombre y Apellido
    j. Mostrar en pantalla las últimas líneas de un archivo.

11. Investigue el funcionamiento, parámetros y ubicación (directorio) de los siguientes

comandos:

- man
- shutdown
- reboot
- halt
- locate
- uname
- dmesg
- lspci
- at
- netstat
- head
- tail

12. Procesos:
    a. ¿Qué es un proceso? ¿A que hacen referencia las siglas PID y PPID? ¿Todos los procesos tienen estos atributos en GNU/Linux? Justifique. Indique qué otros atributos tiene un proceso.
    b. Investigue el funcionamiento, parámetros y ubicación (directorio) de los siguientes comandos relacionados a procesos. En caso de que algún comando no venga por defecto en la distribución que utiliza deberá proceder a instalarlo:

        i. top
        ii. htop
        iii. ps
        iv. pstree
        v. kill
        vi. pgrep
        vii. pkill
        viii. killall
        ix. renice
        x. xkill
        xi. atop
        xii. nice

13. Responda en forma sintética sobre los siguientes conceptos:
    a. ¿Qué es un SO?
    b. Enumere qué componentes/aspectos del Hardware son necesarios para cumplir los objetivos de un SO.
    c. Enumere componentes de un SO
    d. ¿Que es una llamada al sistema (*system call*)? ¿Cómo es posible implementarlas?
    e. Defina Programa y Proceso.
    f. ¿Cuál es la información mínima que el SO debe tener sobre un proceso? ¿En qué estructura de datos asociada almacena dicha información?
    g. ¿Qué objetivos persiguen los algoritmos de planificación (scheduling).
    h. ¿Qué significa que un algoritmo de scheduling sea apropiativo o no apropiativo (Preemptive o Non-Preemptive)?
    i. ¿Qué tareas realizan los siguientes módulos de planificación?:
        i. Short Term Scheduler
        ii. Long Term Scheduler
        iii. Medium Term Scheduler
    j. ¿Qué tareas realiza el Dispatcher? ¿Y el Loader?
    k. ¿Qué significa que un proceso sea “CPU Bound” y “I/O Bound”?
    l. ¿Cuáles son los estados posibles por los que puede atravesar un proceso? ¿Qué

representa que un proceso se encuentre en los estados enumerados? Utilizando un diagrama explique las transiciones entre los estados.

m. ¿Cuáles de los schedulers mencionados anteriormente se encargan de las transiciones entre los estados enumerados?
n. Defina Tiempo de retorno (**TR**) y Tiempo de espera (**TE**) para un proceso.
o. Defina Tiempo Promedio de Retorno (**TPR**) y Tiempo promedio de espera (**TPE**) para un lote de procesos.
p. Defina tiempo de respuesta.

14. Para los siguientes algoritmos de scheduling:
    - FCFS (First Come First Served)
    - SJF (Shortest Job First)
    - Round Robin
    - Prioridades

a. Explique su funcionamiento mediante un ejemplo.
b. ¿Alguno de ellos cuentan con parámetros para su funcionamiento? Identifique y enunciarlos
c. Cual es el más adecuado según los tipos de procesos y/o SO.
d. Cite ventajas y desventajas de su uso.

15. Para el algoritmo Round Robin, existen 2 variantes: **Timer Fijo** y **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?

16. Dado el siguiente lote de procesos en el que todos 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 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.

17. Dado el siguiente lote 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, qué conclusión se puede sacar con respecto al valor del quantum.
   e. ¿Para el algoritmo Round Robin, en qué casos utilizará un valor de quantum alto y qué ventajas y desventajas obtendría?

18. Una variante al algoritmo SJF es el algoritmo SJF apropiativo o SRTF (Shortest Remaining Time First):
   a. Realice el diagrama de Gantt para este algoritmo según el lote de trabajos del ejercicio 5.
   b. ¿Nota alguna ventaja frente a otros algoritmos?

19. Suponga que se agregan las siguientes prioridades al lote de procesos del ejercicio 5, 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 qué circunstancias lo utilizaría y ante qué situaciones considera que la implementación de prioridades podría no ser de mayor relevancia?

20. 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 el inciso b?

21. 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 Kernel 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 más).
Cuando un proceso en ejecución realiza una operación de I/O el mismo es expulsado de la CPU y colocado en la cola correspondiente al dispositivo involucrado en la operación.
El Kernel dispone también de un "I/O Scheduling" que administra 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 5 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 5 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)

22. 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)

23. Para equiparar la desventaja planteada en el ejercicio 10, 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 encuentran 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 faltó 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 9)a) nuevamente considerando este algoritmo, con un quantum de 2 unidades y Timer Variable.

24. 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.

25. 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 fórmula (1):

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

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 primera 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 qué valores se obtendrían como estimación para las ráfagas de CPU del proceso si se utiliza la fórmula 1, con un valor inicial estimado de S_1=10.

La fórmula 1 le da el mismo peso a todos los casos (siempre calcula la media). Es posible reescribirla permitiendo darle un peso mayor a los casos más recientes y menor a casos viejos (o viceversa). Se plantea la siguiente fórmula 2:

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

Con $0 < \alpha < 1$.

b. Analice para qué valores de α se tienen en cuenta los casos más recientes.
c. Para la situación planteada en a) calcule qué valores se obtendrían si se utiliza la fórmula 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 más se asemeja a las ráfagas de CPU reales del proceso?

26. Colas Multinivel

Actualmente 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 dividida en varias colas, teniendo cada una su propio algoritmo de planificación.

a. Suponga que se tiene 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 utilizará 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 qué cola elegir un proceso.

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

27. 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 través 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.

28. 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 está 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 se 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.

Para los casos a y b del ejercicio 15 realice el diagrama de Gantt considerando además que se tiene un envejecimiento de 4 unidades.

29. La situación planteada en el ejercicio 16, 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 qué colas implementaría, qué 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.

30. Ejercicio opcional. Un caso real: “Unix Clásico “ (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 una 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 fórmulas:

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

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

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).

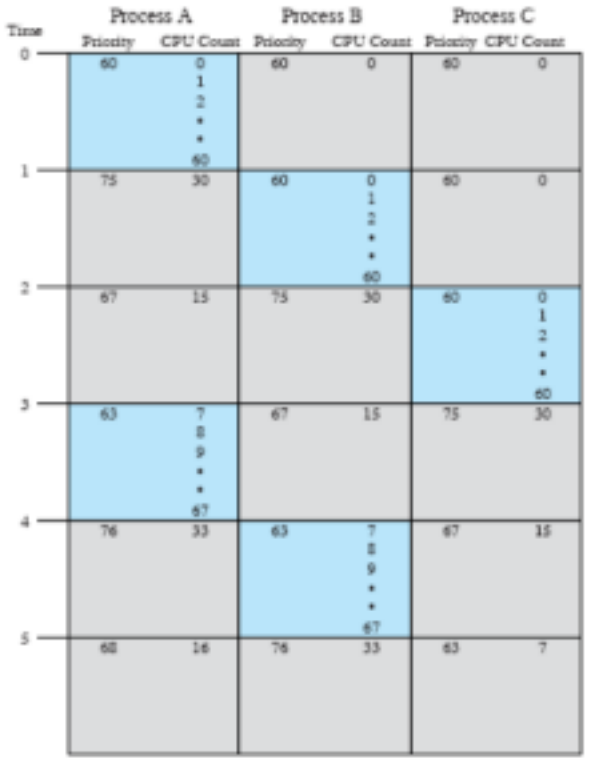
$Base_j$ = Prioridad base del proceso *j*.

$Nice_j$ = 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 descrita 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.

31. A cuáles de los siguientes tipos de trabajos:
    - cortos acotados por CPU
    - cortos acotados por E/S
    - largos acotados por CPU
    - 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.

32. Analizar y explicar por qué si el quantum en Round-Robin se incrementa sin límite, el algoritmo de planificación se aproxima a FIFO.

33. Dado los siguientes programas en un pseudo-código que simula la utilización de llamadas al sistema para crear procesos en Unix/Linux, indicar qué mensajes se imprimirán

en pantalla (sin importar el orden en que saldrían):

a. Caso 1

```
printf("hola")
x = fork()
if x <  1 {
      execv("ls")
      printf("mundo")
      exit(0)
}
exit(0)
```

b. Caso 2

```
printf("hola")
x = fork()
if x <  1 {
      execv("ps")
       printf("mundo")
       exit(0)
}
execv("ls")
printf("fin")
exit(0)
```

c. Caso 3

```
printf("Anda  a rendir el Primer Parcial de Promo!")
newpid = fork()
if newpid == 0 {
    printf("Estoy comenzando el Examen")
    execv("ps")
    printf("Termine el Examen")
}
printf("¿Como te fue?")
exit(0)
printf("Ahora anda a descansar")
```

34. Dado el siguiente programa en C, analice su objetivo sin necesidad de ejecutarlo. (Investigue el funcionamiento del iterador *for* en C para poder responder: https://ccia.ugr.es/~jfv/ed1/c/cdrom/cap4/cap43.htm)

```
#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 ;
}
```

a. ¿Cuántas líneas con la palabra "Proceso" aparecen al final de la ejecución de este programa?

b. *¿El número de líneas es el número de procesos que han estado en ejecución?*. Ejecute el

programa y compruebe si su respuesta es correcta. Modifique el valor del bucle for y compruebe los nuevos resultados.

35. Modifiquemos el programa anterior. Ahora, además de un mensaje, vamos a añadir una variable y antes de finalizar el programa vamos a mostrar el valor de la misma:

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

a. ¿Qué valores se imprimirán en la consola?
b. ¿Todas las líneas tendrán el mismo valor o algunas líneas tendrán valores distintos?
c. ¿Cuál es el valor (o valores) que aparece*?.* Compile y ejecute el programa y compruebe si su respuesta es correcta.
d. Realice modificaciones al programa, por ejemplo: modifique el valor del bucle for, cambie el lugar dónde se incrementa la variable p, y compruebe los nuevos resultados.