**Práctica 4 “Introducción a los Sistemas Operativos”**

***Facultad de Informática, UNLP.***

***Alumno: Gonzalez, Joaquín Manuel.***

***Fecha de Realización:***

Contenido

[***1) Responda en forma sintética sobre los siguientes conceptos:*** 5](#_Toc147742953)

[***(a) Programa y Proceso.*** 5](#_Toc147742954)

[***(b) Defina Tiempo de retorno (TR) y Tiempo de espera (TE) para un Job.*** 5](#_Toc147742955)

[***(c) Defina Tiempo Promedio de Retorno (TPR) y Tiempo promedio de espera (TPE) para un lote de JOBS.*** 5](#_Toc147742956)

[***(d) ¿Qué es el Quantum?*** 6](#_Toc147742957)

[***(e) ¿Qué significa que un algoritmo de scheduling sea apropiativo o no apropiativo (Preemptive o Non-Preemptive)?*** 6](#_Toc147742958)

[***(f) ¿Qué tareas realizan el Short Term, Long Term y Medium Term Scheduler?*** 7](#_Toc147742959)

[***(g) ¿Qué tareas realiza el Dispatcher?*** 7](#_Toc147742960)

[***2) Procesos:*** 7](#_Toc147742961)

[***(a) Investigue y detalle para que sirve cada uno de los siguientes comandos. (Puede que algún comando no venga por defecto en su distribución por lo que deberá instalarlo):*** 7](#_Toc147742962)

[***(b) Observe detenidamente el siguiente código. Intente entender lo que hace sin necesidad de ejecutarlo.*** 10](#_Toc147742963)

[***1. ¿Cuántas líneas con la palabra “Proceso” aparecen al final de la ejecución de este programa?*** 10](#_Toc147742964)

[***2. ¿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.*** 10](#_Toc147742965)

[***(c) 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.*** 12](#_Toc147742966)

[***1. ¿Qué valores se muestran por consola?*** 12](#_Toc147742967)

[***2. ¿Todas las líneas tendrán el mismo valor o algunas líneas tendrán valores distintos?*** 12](#_Toc147742968)

[***3. ¿Cuál es el valor (o valores) que aparece? 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.*** 13](#_Toc147742969)

[***(d) Comunicación entre procesos:*** 13](#_Toc147742970)

[***1. Investigue la forma de comunicación entre procesos a través de pipes.*** 13](#_Toc147742971)

[***2. ¿Cómo se crea un pipe en C?*** 13](#_Toc147742972)

[***3. ¿Qué parámetro es necesario para la creación de un pipe? Explique para que se utiliza.*** 14](#_Toc147742973)

[***4. ¿Qué tipo de comunicación es posible con pipes?*** 14](#_Toc147742974)

[***(e) Cuál es la información mínima que el SO debe tener sobre un proceso? ¿En qué estructura de datos asociada almacena dicha información?*** 14](#_Toc147742975)

[***(f) ¿Qué significa que un proceso sea “CPU Bound” y “I/O Bound”?*** 15](#_Toc147742976)

[***(g) ¿Cuáles son los estados posibles por los que puede atravesar un proceso?*** 15](#_Toc147742977)

[***(h) Explique mediante un diagrama las posibles transiciones entre los estados.*** 16](#_Toc147742978)

[***(i) ¿Que scheduler de los mencionados en 1 f se encarga de las transiciones?*** 16](#_Toc147742979)

[***3. Para los siguientes algoritmos de scheduling: FCFS/FIFO (First Come First Served), SJF (Shortest Job First), Round Robin, Prioridades.*** 17](#_Toc147742980)

[***(a) Explique su funcionamiento mediante un ejemplo.*** 17](#_Toc147742981)

[***(b) ¿Alguno de ellos requiere algún parámetro para su funcionamiento?*** 18](#_Toc147742982)

[***(c) Cual es el más adecuado según los tipos de procesos y/o SO*** 19](#_Toc147742983)

[***(d) Cite ventajas y desventajas de su uso.*** 19](#_Toc147742984)

[***4. Para el algoritmo Round Robin, existen 2 variantes: Timer Fijo y Timer Variable.*** 19](#_Toc147742985)

[***(a) ¿Qué significan estas 2 variantes?*** 19](#_Toc147742986)

[***(b) Explique mediante un ejemplo sus diferencias.*** 19](#_Toc147742987)

[***(c) En cada variante ¿Dónde debería residir la información del Quantum?*** 20](#_Toc147742988)

[***5. Se tiene el siguiente lote de procesos que arriban al sistema en el instante 0 (cero):*** 20](#_Toc147742989)

[***(a) Realice los diagramas de Gantt según los siguientes algoritmos de scheduling:*** 21](#_Toc147742990)

[***(b) Para cada algoritmo calcule el TR y TE para cada job así como el TPR y el TPE.*** 21](#_Toc147742991)

[***(c) En base a los tiempos calculados compare los diferentes algoritmos.*** 21](#_Toc147742992)

[***6. Se tiene el siguiente lote de procesos:*** 21](#_Toc147742993)

[***(a) Realice los diagramas de Gantt según los siguientes algoritmos de scheduling:*** 21](#_Toc147742994)

[***(b) Para cada algoritmo calcule el TR y TE para cada job así como el TPR y el TPE.*** 22](#_Toc147742995)

[***(c) En base a los tiempos calculados compare los diferentes algoritmos.*** 22](#_Toc147742996)

[***(d) En el algoritmo Round Robin, que conclusión se puede sacar con respecto al valor del quantum.*** 22](#_Toc147742997)

[***(e) ¿Para el algoritmo Round Robin, en qué casos utilizaría un valor de quantum alto y que ventajas y desventajas obtendría?*** 22](#_Toc147742998)

[***7. Una variante al algoritmo SJF es el algoritmo SJF apropiativo o SRTF (Shortest Remaining Time First):*** 23](#_Toc147742999)

[***(a) Realice el diagrama del Gantt para este algoritmo según el lote de trabajos del ejercicio 6.*** 23](#_Toc147743000)

[***(b) ¿Nota alguna ventaja frente a otros algoritmos?*** 23](#_Toc147743001)

[***8. Suponga que se agregan las siguientes prioridades al lote de procesos del ejercicio 6, donde un menor número indica mayor prioridad:*** 23](#_Toc147743002)

[***(a) Realice el diagrama de Gantt correspondiente al algoritmo de planificación por prioridades según las variantes: Apropiativo y No Apropiativo.*** 24](#_Toc147743003)

[***(c) ¿Nota alguna ventaja frente a otros algoritmos? Bajo qué circunstancias lo utilizaría y ante que situaciones considera que la implementación de prioridades podría no ser de mayor relevancia?*** 24](#_Toc147743004)

[***9. Inanición (Starvation)*** 24](#_Toc147743005)

[***(a) ¿Qué significa?*** 24](#_Toc147743006)

[***(b) ¿Cuál/es de los algoritmos vistos puede provocarla?*** 24](#_Toc147743007)

[***(c) ¿Existe alguna técnica que evite la inanición para el/los algoritmos mencionados en b?*** 25](#_Toc147743008)

[***10. Los procesos, durante su ciclo de vida, pueden realizar operaciones de I/O como lecturas o escrituras a disco, cintas, uso de impresoras, etc.*** 25](#_Toc147743009)

[***(a) Suponga que al lote de procesos del ejercicio 6 se agregan las siguientes operaciones de entrada salida:*** 25](#_Toc147743010)

[***(b) Suponga que al lote de procesos del ejercicio 6 se agregan las siguientes operaciones de entrada salida:*** 26](#_Toc147743011)

# ***1) Responda en forma sintética sobre los siguientes conceptos:***

## ***(a) Programa y Proceso.***

* **Definiciones:**
* **Programa:** Conjunto de instrucciones o código escrito en un lenguaje de programación que especifica una serie de acciones a realizar por una computadora para llevar a cabo una tarea específica.
* **Es estático.**
* **No tiene program counter.**
* **Existe desde que se edita hasta que se borra.**
* **Proceso:** Instancia en ejecución de un programa en un sistema operativo.
* **Es dinámico.**
* **Tiene program counter.**
* **Su ciclo de vida comprende desde que se lo ejecuta hasta que termina.**
* **Poseen estados y en todo momento están en uno de ellos.**

## ***(b) Defina Tiempo de retorno (TR) y Tiempo de espera (TE) para un Job.***

* **Definiciones:**
* **Tiempo de Retorno (TR):** Tiempo que transcurre entre que el proceso llega al sistema hasta que completa su ejecución.
* **Tiempo de Espera (TE):** Tiempo que el proceso se encuentra en el sistema esperando, es decir, el tiempo que pasa sin ejecutarse **(TR – Tiempo de CPU “Tcpu”).**

## ***(c) Defina Tiempo Promedio de Retorno (TPR) y Tiempo promedio de espera (TPE) para un lote de JOBS.***

* **Definiciones:**
* **Tiempo Promedio de Retorno (TPR):** Es el promedio de los tiempos de retorno de todos los procesos del lote **(suma de TR de cada proceso del lote / cantidad de procesos del lote).**
* **Tiempo Promedio de Espera (TPE):** Es el promedio de los tiempos de espera de todos los procesos del lote **(suma de TE de cada proceso del lote / cantidad de procesos del lote).**

## ***(d) ¿Qué es el Quantum?***

* Es la medida que determina cuanto tiempo podrá usar el procesador cada proceso.

**Pueden ser de dos tamaños:**

* **Pequeño:** Genera un alto costo de context switch.
* **Grande:** Un proceso puede bloquear la CPU por mucho tiempo, pudiendo llegar a reducir la capacidad de respuesta del sistema ante otras tareas.

## ***(e) ¿Qué significa que un algoritmo de scheduling sea apropiativo o no apropiativo (Preemptive o Non-Preemptive)?***

* **Definiciones:**
* **Algoritmos Apropiativos (Preemptive):** El proceso que esté en ejecución en la CPU puede ser interrumpido/expulsado y llevado a la cola de listos dependiendo los criterios del algoritmo.
* **Mayor overhead (costo adicional) pero mejor servicio.**
* **Los procesos no monopolizan el procesador.**
* **Algoritmos No Apropiativos (Non-Preemptive):** Una vez que un proceso alcanza el estado “running”, continua hasta que termina o se bloquea por algún evento (por ejemplo I/O) y no puede ser expulsado/interrumpido.

## ***(f) ¿Qué tareas realizan el Short Term, Long Term y Medium Term Scheduler?***

* **Short Term Scheduler:** Selecciona/Determina que proceso pasará a ejecutarse, es decir, se encarga de la selección de los procesos que están en estado **“ready to run”** que pasarán a estado **“running”.**
* **Long Term Scheduler:** Admite nuevos procesos a memoria controlando el grado de multiprogramación, es decir, es el encargado de seleccionar los procesos en estado **“new”** que serán admitidos pasando a estado **“ready to run”.**
* **Medium Term Scheduler:** Encargado de realizar el **swapping**, es decir, el intercambio entre el disco y la memoria **(swapin)** y viceversa **(swapout)** cuando el sistema operativo lo determina, esta acción puede disminuir el grado de multiprogramación. **Utiliza la partición SWAP.**

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

* Se encarga de realizar el **context switch** (guardar el estado del proceso actual que se está ejecutando y cargar el estado del próximo procesos a ejecutarse) y **cambio del modo de ejecución**. Es utilizado por el **Short Term Scheduler** ya que este se encarga de seleccionar los procesos pero quien realiza el cambio del proceso de “ready to run” a “running” es el **Dispatcher.**

# ***2) Procesos:***

## ***(a) Investigue y detalle para que sirve cada uno de los siguientes comandos. (Puede que algún comando no venga por defecto en su distribución por lo que deberá instalarlo):***

* **Comandos:**
* **“top”:** Herramienta interactiva que proporciona una actualización en tiempo real de los procesos que están usando la CPU y la memoria.
* **“htop”:** Herramienta que nos muestra información detallada y en tiempo real sobre los procesos en ejecución y la utilización de recursos del sistema de una forma más amigable que el comando **top.** Podremos ver los procesos en lista, el árbol de procesos, utilizar funciones de búsqueda y filtrado, información sobre el sistema y la utilización de los recursos, todo esto dentro de una interfaz interactiva. **Se debe instalar con “apt-get install htop”.**
* **“ps”:** Muestra información sobre los procesos en ejecución. **(PID, TTY “Terminal”, TIME “tiempo de ejecución” y CMD “comando”).**

**Parámetros:**

* **“-e”:** Muestra todos los procesos del sistema.
* **“-aux”:** Proporciona una vista detallada de todos los procesos del sistema.
* **“pstree”:** Muestra una representación gráfica de la jerarquía de procesos.
* **“kill”:** Se utiliza para enviar señales a procesos en ejecución, permite la interacción con los procesos, como terminarlos de manera controlada o enviarles señales específicas. Requiere el uso del PID.

**Señales:**

* **SIGTERM:** Si solo se especifica el PID del proceso a terminar, se envía esta señal que es una petición suave para terminar el proceso.
* **SIGKILL:** Si se utiliza el comando con el parámetro **“-9 PID”** se envía una señal fuerte de terminación que hará que se termine el proceso de inmediato sin darle chance de realizar acciones de limpieza o un cierre ordenado.
* **“pgrep”:** Busca y lista los PID de los procesos que coinciden con un criterio de búsqueda específico. **“pgrep [opciones] [patrón]”**

**Parámetros:**

* **“-l”:** Muestra el nombre del proceso junto con su PID.
* **“-f”:** Busca el patrón en la línea de comando completa del proceso.
* **“-u”:** Filtra por el nombre de usuario que inició el proceso.
* **“-x”:** Realiza una coincidencia exacta con el patrón.
* **“pkill”:** Similar a **pgrep**, aplica la búsqueda de procesos pero a estos les hace las funcionalidades del comando **kill. “pkill [opciones] nombre\_del\_proceso”.**

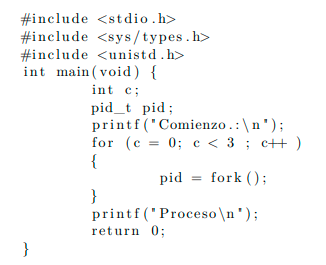
**Parámetros:**

* **“-u [usuario]”:** Permite especificar el nombre del usuario a cuyos procesos se le quieren enviar señales.
* **“-x”:** Realiza una coincidencia exacta con el nombre del proceso.
* **“-f”:** Fuerza la terminación.
* **“-g [group]”:** Termina los procesos que pertenecen a un grupo específico.
* **“pkillall”:** Igual a **pkill** pero este permite enviar señales a procesos por su nombre o por su PID.
* **“killall”:** Permite terminar procesos basados en sus nombres, a diferencia de kill que requiere del PID.

**Parámetros:**

* **“-u”:** Especificar el nombre del usuario cuyos procesos se deben terminar.
* **“-s”:** Especificar la señal que se enviará a los procesos.
* **“-l”:** Lista todas las señales disponibles.
* **“-g”:** Termina todos los procesos pertenecientes un grupo de procesos.
* **“renice”:** Herramienta utilizada para ajustar la prioridad de ejecución de un proceso en tiempo real, normalmente la prioridad se representa como un valor numérico que oscila entre -20 y +19 y mientras más bajo sea el valor, mayor prioridad.
* **“xkill”:** Utilidad que se utiliza para terminar de manera forzada una aplicación gráfica que no responde o está causando problemas.
* **“atop”:** Herramienta interactiva que funciona de manera similar a **top** pero que nos muestra la carga de trabajo dentro del sistema. Indica la ocupación de los recursos más importantes del hardware a nivel del sistema y qué procesos son responsables de la carga indicada de CPU y memoria.

## ***(b) Observe detenidamente el siguiente código. Intente entender lo que hace sin necesidad de ejecutarlo.***

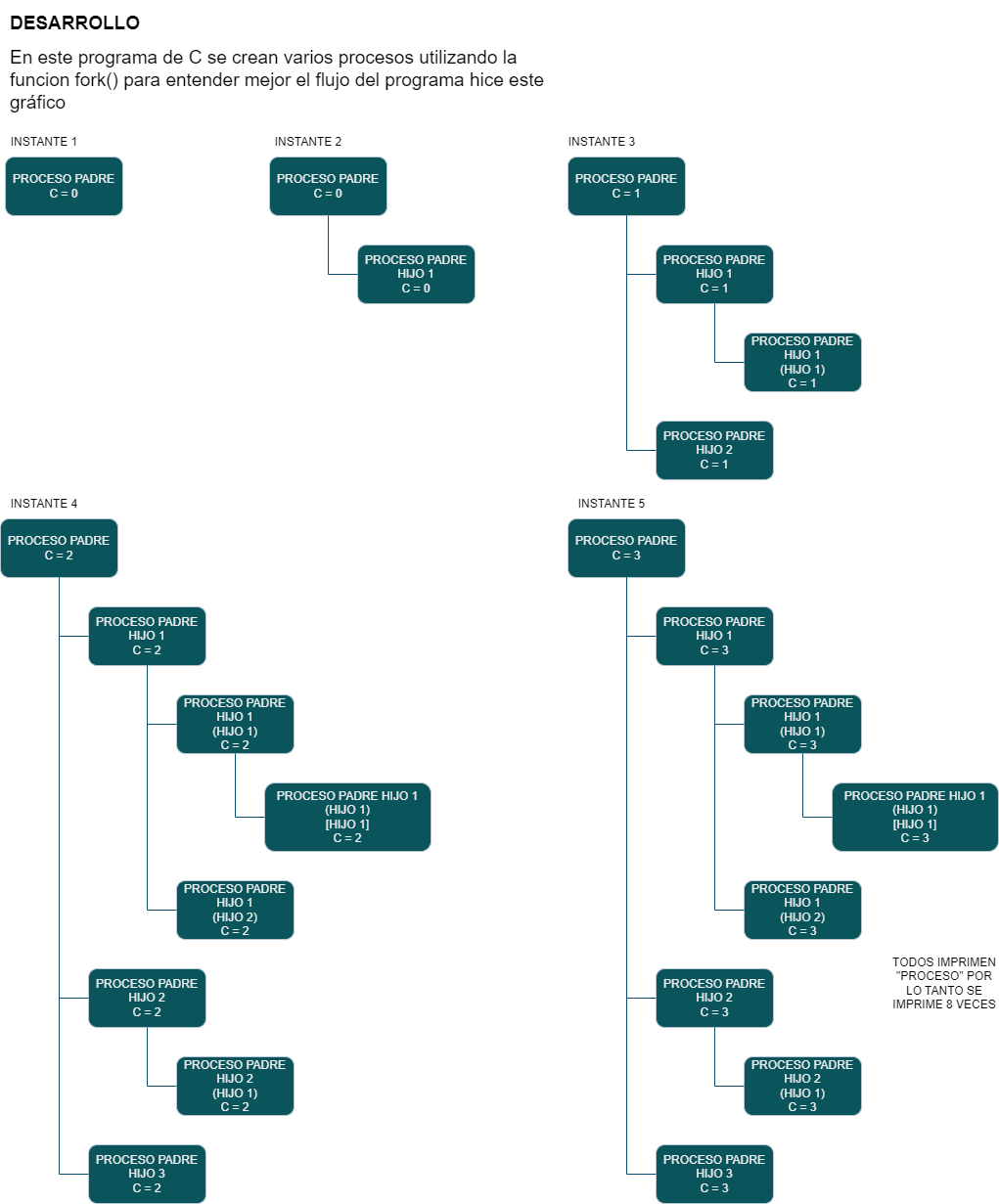


### ***1. ¿Cuántas líneas con la palabra “Proceso” aparecen al final de la ejecución de este programa?***

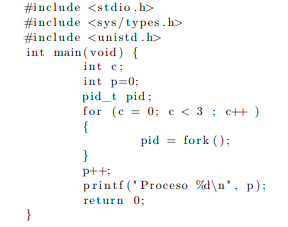
* Al final de la ejecución del programa aparecen 8 líneas con la palabra **“Proceso”.**

### ***2. ¿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.***

* Si, el número de líneas que imprimen la palabra **“Proceso”** son la cantidad de procesos que se crean. Si ejecutamos el programa modificando el valor del bucle for vamos a ver que la cantidad de procesos crece exponencialmente en potencias de 2 (con c = 1 tenemos 2 procesos, con c = 2 tenemos 4, con c = 3 tenemos 8, con c = 4 tenemos 16; etc).



## ***(c) 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.***



### ***1. ¿Qué valores se muestran por consola?***

* Por consola se muestran 8 líneas con la palabra **“Proceso1”** ya que el valor de la variable **“p”** cada vez que se hace un **“fork()”** es 0 por lo tanto los 8 procesos que se crean ya que la variable **“c”** si va cambiando de valor a medida que se hacen los **“fork()”,** incrementan una única vez el valor de la variable **“p”** al final del código dejándola con valor 1 e imprimiendo su contenido.

### ***2. ¿Todas las líneas tendrán el mismo valor o algunas líneas tendrán valores distintos?***

* Todas las líneas tendrán el mismo valor.

### ***3. ¿Cuál es el valor (o valores) que aparece? 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.***

* Ya vimos lo que pasa si modificamos el valor del bucle for, ahora, si hacemos que el valor de **“p”** se incremente antes o después del bucle for, el resultado va a ser el mismo, pero si hacemos que la variable **“p”** se incremente dentro del bloque for el resultado será 8 líneas con **“Proceso3”.**

## ***(d) Comunicación entre procesos:***

### ***1. Investigue la forma de comunicación entre procesos a través de pipes.***

* Un **Pipe ( | )** permite la comunicación entre procesos al redirigir la salida estándar de un proceso hacia la entrada estándar de otro.

**Tiene 2 finalidades:**

* **Procesamiento en tiempo real:** Permite el procesamiento en tiempo real de la salida de un comando antes de ser utilizada por otro.
* **Combinación de comandos:** Facilita la comunicación entre comandos para lograr tareas más complejas.

### ***2. ¿Cómo se crea un pipe en C?***

* Para crear un **Pipe** en C lo podemos hacer mediante la función **“pipe()”** que se encuentra en la biblioteca de manejo de archivos **“unistd.h”,** además es necesario manejar un arreglo de enteros de dos posiciones para almacenar los **descriptores de archivo** (números que el sistema operativo usa para identificar y manipular archivos y canales de comunicación como los pipes) donde en la primera se usará para hacer referencia al **extremo de lectura** del pipe y la segunda posición hará referencia al **extremo de escritura**. **“pipe(fd[N]) siendo N 0 ó 1 y fd un arreglo de enteros”.**

### ***3. ¿Qué parámetro es necesario para la creación de un pipe? Explique para que se utiliza.***

* Como fue mencionado antes, la función **“pipe()”** necesita como parámetro un arreglo de enteros de dos posiciones, este es importante ya que es la forma en que se comunica al programa dónde se encuentran los extremos del **pipe** para comunicar los procesos. Es necesario también combinar el uso de pipes con la función **“close()”** que se encarga de cerrar **descriptores de archivos** haciendo que se liberen recursos de memoria, que haya una prevención de fugas de recursos y que no haya bloqueos entre los canales de comunicación.

### ***4. ¿Qué tipo de comunicación es posible con pipes?***

* Con los **Pipes** es posible lograr la comunicación entre procesos, estos son **unidireccionales**, es decir, un proceso puede escribir o leer un pipe, pero no ambas a la vez, y además, solo podría hacerlo un proceso padre a su hijo por ejemplo. Aunque sean **unidireccionales,** se puede combinar el uso de dos **pipes** para crear un canal **bidireccional.**

## ***(e) Cuál es la información mínima que el SO debe tener sobre un proceso? ¿En qué estructura de datos asociada almacena dicha información?***

* La información mínima que el sistema operativo debe tener sobre un proceso es el **PID,** el **PPID**, la **Ubicación del mismo en memoria,** los **Recursos Asignados (espacio de memoria, descriptores de archivos, etc.),** el **Accounting,** la **Entrada y Salida,** el **Contexto de Ejecución (valores de los registros de la CPU),** y la **Planificación del mismo (prioridad, estado, tiempo consumido, etc.).**

Toda esta información se almacena en una estructura llamada **PCB (Process Control Block).**

**Características de la PCB:**

* Existe una por proceso.
* Es lo primero que se crea cuando un proceso es creado y también es lo último que se borra cuando el proceso termina.
* Tiene referencias a memoria.
* Se puede pensar como un gran registro que almacena toda esta información.

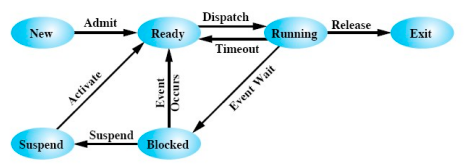
## ***(f) ¿Qué significa que un proceso sea “CPU Bound” y “I/O Bound”?***

* **CPU Bound:** Un proceso es **CPU Bound** cuando su tiempo de ejecución está ligado y limitado por la CPU, normalmente son procesos que realizan tareas intensivas y consumen una gran cantidad de ciclos de CPU.
* **I/O Bound:** Un proceso es **I/O Bound**  cuando su tiempo de ejecución está ligado y limitado por la entrada/salida, normalmente pasan la mayor pate e su tiempo esperando a que las operaciones de I/O se completen en vez de realizar tareas intensivas sin consumir una gran cantidad de ciclos de CPU.

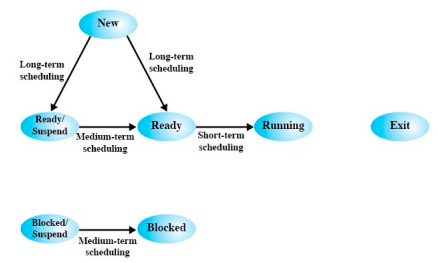
## ***(g) ¿Cuáles son los estados posibles por los que puede atravesar un proceso?***

* **Estados:**
* **“New”:** En este estado se crea la PCB y se carga al proceso en memoria dejando al proceso en un estado de inicialización.
* **“Ready to run”:** El proceso está listo para cargar su Program Counter y empezar a ejecutarse, pero no lo está haciendo, se encuentra almacenado en la cola de listos compitiendo por la CPU.
* **“Running”:** Se genera un cambio de contexto y el proceso empieza su ejecución teniendo la CPU a su disposición hasta que se termine su Quantum, se necesite una operación de I/O, hasta que termine o hasta que sea bloqueado. Puede tener tres salidas: **“Exit”, “Waiting” o “Ready to run”.**
* **“Exit”:** Se libera la CPU y se eliminan todas las estructuras del proceso en la memoria.
* **“Waiting”:** El proceso pasa a este estado cuando necesita esperar que se cumpla un evento para seguir ejecutándose (System calls, I/O, etc.). Queda en un estado de bloqueo hasta que la espera termina.

## ***(h) Explique mediante un diagrama las posibles transiciones entre los estados.***

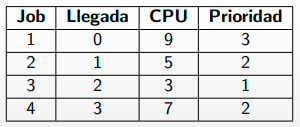


## ***(i) ¿Que scheduler de los mencionados en 1 f se encarga de las transiciones?***



# ***3. Para los siguientes algoritmos de scheduling: FCFS/FIFO (First Come First Served), SJF (Shortest Job First), Round Robin, Prioridades.***

## ***(a) Explique su funcionamiento mediante un ejemplo.***



* **FCFS:**
* Cuando hay que elegir un proceso para ejecutar, se elige al más viejo.
* No favorece a ningún tipo de procesos.
* Los CPU Bound terminan al comenzar su primer ráfaga mientras que los I/O Bound no.
* Es no apropiativo.

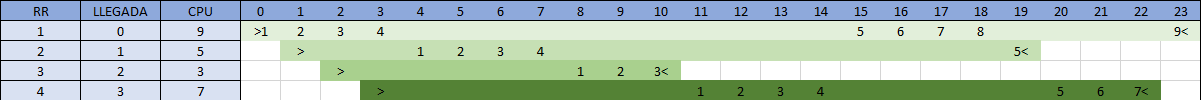
**Orden de ejecución: Job 1, Job 2, Job 3, Job 4.**

* **SJF:**
* Selecciona al proceso con la ráfaga más corta.
* Es no apropiativo.
* Los procesos cortos y los I/o Bound se ven beneficiados.
* Los procesos largos pueden sufrir starvation (inanición) ya que pueden pasar mucho tiempo sin ser seleccionados.

**Orden de ejecución: Job 1, Job 3, Job 2, Job 4.**

* **Round Robin:**
* Política basada en el uso de un reloj (Quantum).
* Cuando un proceso es expulsado de la CPU es colocado al final de la cola de listos y se selecciona otro generando un **FIFO circular.**
* Existe un “contador” que indica las unidades de CPU en las que el proceso se ejecutó. Cuando este llega a 0 el proceso es expulsado.
* El “contador” puede ser Global o Local (PCB).
* Se puede tener 2 variantes con respecto al valor inicial del “contador” cuando un proceso es asignado a la CPU: **Timer Variable o Timer Fijo.**

**Orden de ejecución con un Quantum = 4 y Timer Variable:**



* **Prioridades:**
* Cada proceso tiene un valor que representa su prioridad, mientras menor sea su valor, mayor prioridad.
* Se selecciona al proceso con mayor prioridad de la cola de listos.
* Puede ser apropiativo o no.
* Existe una cola de listos para cada nivel de prioridad.
* Los procesos de baja prioridad pueden sufrir starvation (inanición).

**Orden de ejecución No Apropiativo: Job 1, Job 3, Job 2, Job 4.**

## ***(b) ¿Alguno de ellos requiere algún parámetro para su funcionamiento?***

* Únicamente el **Round Robin** necesita la definición del tamaño del Quantum.

## ***(c) Cual es el más adecuado según los tipos de procesos y/o SO***

* **Algoritmos:**
* **FIFO/FCFS:** No favorece a ningún tipo de procesos.
* **SJF:** Favorece a los I/O Bound y a los procesos cortos.
* **Round Robin:** Favorece más a los I/O Bound.
* **Prioridades:** Favorece a los de mayor prioridad.

## ***(d) Cite ventajas y desventajas de su uso.***

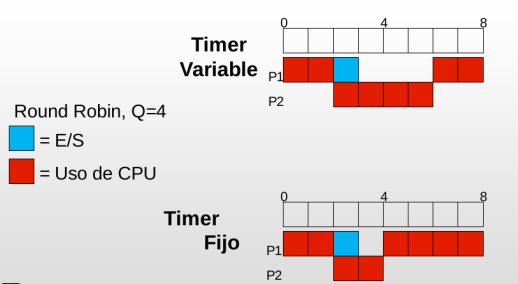
* TODO EXPLICADO MÁS ARRIBA.

# ***4. Para el algoritmo Round Robin, existen 2 variantes: Timer Fijo y Timer Variable.***

## ***(a) ¿Qué significan estas 2 variantes?***

* **Timer Variable:** El “contador” se inicializa en Q (contador = Q) cada vez que un proceso es asignado a la CPU, es decir, si un proceso no llega a usar todo su Quantum, el sobrante se tira a la basura y no se reutiliza.
* **Timer Fijo:** El “contador” se inicializa en Q (contador = Q) cuando su valor es cero, es decir, si un proceso no llega a usar todo su Quantum, el sobrante lo usará el siguiente proceso a ejecutarse y cuando el Quantum llegue a cero, se reasignará su valor.

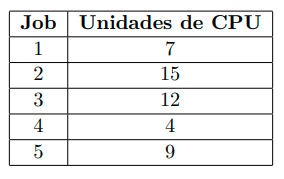
## ***(b) Explique mediante un ejemplo sus diferencias.***



## ***(c) En cada variante ¿Dónde debería residir la información del Quantum?***

* Para el **Timer Variable,** cada proceso debería almacenar la información de su Quantum en su PCB, mientras que con **Timer Fijo**, la información se debe almacenar en una estructura o espacio que sea accesible directamente por el sistema operativo.

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



## ***(a) Realice los diagramas de Gantt según los siguientes algoritmos de scheduling:***

* Ver carpeta de ejercicios resueltos.

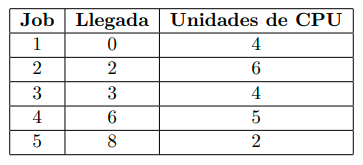
## ***(b) Para cada algoritmo calcule el TR y TE para cada job así como el TPR y el TPE.***

* Ver carpeta de ejercicios resueltos.

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

* Para este lote de procesos, el algoritmo con mejores promedios fue el **SJF** y el que tuvo peores promedios fue el **Round Robin de Timer Variable con Quantum 4.**

# ***6. Se tiene el siguiente lote de procesos:***



## ***(a) Realice los diagramas de Gantt según los siguientes algoritmos de scheduling:***

* Ver carpeta de ejercicios resueltos.

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

* Ver carpeta de ejercicios resueltos.

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

* El algoritmo con mejores promedios fue el **SJF**, mientras que el algoritmo con peor promedio fue el **Round Robin de Timer Variable con Quantum 1.**

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

* Respecto al valor del Quantum podemos decir que mientras más chico sea, más cambios de contexto habrá, haciendo que los promedios se disparen, y cuanto más grande sea, se comporta de manera similar al **FCFS** disminuyendo los promedios.

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

* Yo utilizaría un Quantum alto para manejar lotes de procesos CPU Bound ya que estos en lo general ocupan por un largo tiempo los recursos de la CPU haciendo que seguramente se amolden bien al valor del Quantum, disminuyendo las tasas de tiempo de retorno de todos estos procesos, factor que toma mucha importancia cuando hablamos de procesos CPU Bound. La desventaja de esto sería que si en el lote de procesos nos encontramos procesos cortos o por ejemplo procesos I/O Bound que por lo general no hacen mucho consumo de la CPU, con un Quantum alto podríamos generar que las tasas de tiempo de espera se disparen, haciendo que los procesos cortos tengan que esperar muchos ciclos de CPU para poder ejecutarse.

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

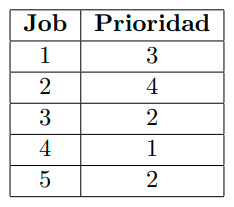
## ***(a) Realice el diagrama del Gantt para este algoritmo según el lote de trabajos del ejercicio 6.***

* Ver carpetas de ejercicios resueltos.

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

* Genera un buen promedio de tiempo de espera, por lo tanto, favorece ventajosamente a procesos I/O Bound ya que estos dependen del tiempo de espera, también los procesos cortos se verían beneficiados.

# ***8. Suponga que se agregan las siguientes prioridades al lote de procesos del ejercicio 6, donde un menor número indica mayor prioridad:***



## ***(a) Realice el diagrama de Gantt correspondiente al algoritmo de planificación por prioridades según las variantes: Apropiativo y No Apropiativo.***

* Ver carpetas de ejercicios resueltos.

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

* Las ventajas que les puedo dar a estos algoritmos es que para tareas críticas del sistema operativo que necesiten ser ejecutadas inmediatamente, van a ser muy buenos, además de que nos brindan una mayor adaptabilidad en nuestro sistema si decidimos llevar una organización por prioridades. Los casos donde los utilizaría son justamente los mencionados en las ventajas.

# ***9. Inanición (Starvation)***

## ***(a) ¿Qué significa?***

* Cuando un proceso sufre de Starvation/Inanición significa que está en una situación donde no puede avanzar o ejecutarse debido a la competencia por recursos del sistema entre diferentes procesos. Ocurre cuando un proceso no puede obtener los recursos necesarios para su ejecución porque otros están acaparándolos.

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

* De los algoritmos vistos los que pueden generar inanición serían el **SJF**, el **STRF** y el de **Prioridades**. Los 2 primeros generándola para procesos largos y el tercero en procesos de baja prioridad.

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

* Existen varias técnicas para evitar la Inanición, algunas de ellas son:
* **Envejecimiento (Aging):** Consiste en aumentar la prioridad de un proceso a medida que pasa el tiempo sin ser seleccionado para su ejecución.
* **Establecimiento de límites de espera:** Consiste en establecer límites de tiempo máximo para la espera de un proceso en la cola de listos, haciendo que si llega a este límite, pase a ejecutarse.
* **Prioridades dinámicas:** Podemos tener sistemas que ajusten las prioridades de los procesos dinámicamente según el comportamiento y las necesidades del proceso.

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

## ***(a) Suponga que al lote de procesos del ejercicio 6 se agregan las siguientes operaciones de entrada salida:***

* Ver carpeta de ejercicios resueltos.

## ***(b) Suponga que al lote de procesos del ejercicio 6 se agregan las siguientes operaciones de entrada salida:***

* Ver carpeta de ejercicios resueltos.