SISTEMAS OPERATIVOS (Código: 71902048) **Septiembre 2024**

Material permitido: **Solo calculadora no programable** Tiempo: **120 minutos** **N**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

1. Conteste **razonadamente** a las siguientes preguntas:

I) *(1 p)* Enumerar y explicar las cuatro *condiciones* necesarias y suficientes para la *existencia de interbloqueo*.

II) *(1 p)* Señalar las principales ventajas e inconvenientes de la técnica de *particionamiento dinámico* de la memoria principal frente a la técnica de *particionamiento fijo*.

III) *(1 p)* Describir el funcionamiento de la técnica de *paginación por demanda*.

IV) *(1 p)* Enumerar y describir las tres principales formas de *implementación de directorios* en un sistema de archivos.

2. *(2 p)* Describir brevemente los *principales estados* en los que se puede encontrar un determinado proceso y dibujar el *diagrama de transición de estados*, **adecuadamente rotulado**, de un sistema operativo que soporte los estados anteriormente descritos.

3. Considérense los procesos A, B, C y D cuyo tiempo de llegada, prioridad y tiempo de servicio se muestran en la Tabla 1.

Proceso	Tiempo de llegada (ut)	Prioridad	Tiempo de servicio (ut)
A	0	1	4
B	0	2	2
C	1	3	3
D	1	4	5

Tabla 1

Supuesto que 1 es la prioridad más alta, que el tiempo de colocación en la cola de procesos preparados es despreciable, que el tiempo de *cambio de contexto* es de 1 ut y que el sistema operativo utiliza un algoritmo de planificación de *turno rotatorio* con un cuanto $q = 2$ ut. Suponer que si varios procesos tienen el mismo tiempo de llegada se colocan en la cola de procesos preparados por orden de prioridad. Se pide:

a) *(0.7 p)* Representar el diagrama de uso del procesador.

b) *(0.3 p)* Determinar el tiempo de retorno y el tiempo de espera de cada proceso.

c) *(1 p)* Repetir los dos apartados anteriores suponiendo que el sistema operativo utiliza un algoritmo de planificación basado en *prioridades de tipo expropiativo*.

Material permitido: **Solo calculadora no programable** Tiempo: **120 minutos** **N**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

4. *(2 p)* En una oficina municipal de atención al ciudadano existen 4 ventanillas. Cuando un ciudadano entra en la oficina para realizar alguna gestión debe guardar una única cola hasta que alguna ventanilla queda libre. Escribir el pseudocódigo de un programa basado en C que usando un **monitor** coordine la actividad de los ciudadanos en la oficina. Considerar la solución de Hansen en el comportamiento de la operación `signal_mon`. El programa debe tener tres partes: pseudocódigo del monitor, pseudocódigo del proceso ciudadano y pseudocódigo de la función principal para lanzar la ejecución concurrente de los procesos.

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Septiembre 2024

Solución Ejercicio 1

I) Para que se produzca un interbloqueo se deben cumplir necesariamente las siguientes cuatro condiciones:

1. *Exclusión mutua*. Cada instancia de un recurso solo puede ser asignada a un proceso como máximo.
2. *Retención y espera*. Cada proceso retiene los recursos que le han sido asignado mientras espera por la adquisición de los otros recursos que necesita.
3. *No existencia de expropiación*. Si un proceso posee un recurso, éste no se le puede expropiar.
4. *Espera circular*. Existe una cadena circular de dos o más procesos, de tal forma que cada proceso de la cadena se encuentra esperando por un recurso retenido por el siguiente proceso de la cadena.

II) La técnica de particionamiento dinámico presenta las siguientes ventajas con respecto a la técnica de particionamiento fijo:

- *Minimiza la fragmentación interna*.La fragmentación interna de una partición es prácticamente despreciable, ya que como máximo será cercana a una unidad de asignación.
- *Permite ejecutar procesos de mayor tamaño*. En un determinado instante de tiempo el tamaño máximo del proceso que puede cargarse en memoria viene limitado por la existencia de una hueco de tamaño igual o mayor que lo pueda contener. Inicialmente en la memoria solo está cargado el sistema operativo, por lo que el resto de la memoria formaría un hueco de tamaño S que podría albergar un proceso de tamaño máximo menor o igual a S.
- *Permite asignar fácilmente más espacio a procesos cuya región de datos o pila aumenta por encima del tamaño inicialmente asignado*. Para ello el sistema operativo puede proceder de dos formas. Una opción es crear una partición de mayor tamaño y reubicar al proceso en dicha partición. Otra posibilidad es añadir a la partición del proceso el espacio de un hueco adyacente a la partición.

Por otra parte la técnica del particionamiento dinámico presenta las siguientes desventajas con respecto a la técnica de particionamiento fijo:

- *Las estructuras de datos que requiere para su implementación y los algoritmos para su gestión son más complejos*. En consecuencia se produce una mayor sobrecarga al sistema.
- *Produce fragmentación externa*. Lo que reduce el aprovechamiento de la memoria.

III) La técnica de *paginación por demanda*, también conocida como técnica de *demanda de página*, al igual que la técnica de paginación simple divide el espacio de la memoria principal en bloques de igual tamaño denominados marcos de página. Además divide el espacio de un proceso en bloques del mismo tamaño denominados páginas. En un determinado instante de tiempo un marco de página *j* puede estar libre o ocupado por una página *i* de un proceso X.

En la paginación por demanda, a diferencia de la paginación simple, para que un proceso se pueda ejecutar no es necesario que todas las páginas del mismo estén cargadas en memoria principal, únicamente se cargan las páginas que se van referenciando durante la ejecución del proceso. Cuando se referencia a una página que no está cargada en la memoria principal el hardware produce una excepción denominada *fallo de página*. El sistema operativo se encarga de atender los fallos de

página. Básicamente el tratamiento de un fallo de página requiere que se busque un marco libre o se elija uno ocupado para ser reemplazado, y se copie en el mismo desde memoria secundaria la página que produjo el fallo de página.

IV) Las tres principales formas de implementación de directorios en un sistema de archivos son:

- *Directorios con entradas de igual tamaño*. Todas las entradas de un directorio poseen el mismo tamaño. En cada entrada de un directorio primero se almacenan las informaciones asociadas al archivo (atributos o número de nodo-i) y a continuación el nombre del archivo.
- *Directorios con entradas de tamaño variable*. Cada entrada del directorio tiene un determinado tamaño. En cada entrada de un directorio primero se almacena el tamaño que ocupa la entrada. A continuación se almacenan las informaciones asociadas al archivo (atributos o número de nodo-i) y finalmente se almacena el nombre del archivo que no puede superar un cierto tamaño máximo. Al nombre del archivo se le añade un carácter especial, por ejemplo un 0, para marcar el final del nombre. Además se añaden algunos caracteres de relleno para que el tamaño de una entrada sea un número entero positivo de palabras de memoria principal. De este modo se evita que dentro de una misma palabra haya información sobre dos entradas de directorios distintas, lo que complicaría la administración de los directorios.
- *Directorios con entradas de igual tamaño y uso de un montículo (heap) para almacenar los nombres de los archivos*. Cada entrada del directorio tiene el mismo tamaño, en ella se almacena un puntero al comienzo del nombre archivo dentro del montículo y las informaciones asociadas al archivo (atributos o número de nodo-i). Al final del directorio se almacena la estructura de datos de tipo montículo para almacenar de nombres de archivos. Dentro del montículo los nombres se almacenan de forma contigua incluyendo un carácter especial para marcar el final de un nombre de archivo.

Solución Ejercicio 2

Aunque el número de estados y su nombre depende de cada sistema operativo, algunos de los estados más habituales en que puede encontrarse un proceso son los siguientes:

- *Nuevo*. El proceso acaba de ser creado pero todavía no se encuentra preparado para ser ejecutado, puesto que aunque se le han asignado algunas estructuras de datos aún no se encuentra cargado en la memoria principal.
- *Preparado*. El proceso está listo para ser ejecutado tan pronto como el planificador del sistema operativo lo considere oportuno.
- *Ejecutándose*. El proceso está siendo ejecutado en el procesador. En un computador con un único procesador solo puede existir en un determinado instante de tiempo un único proceso en este estado.
- *Bloqueado*. El proceso tiene que esperar hasta que se produzca un determinado evento, como por ejemplo, la finalización de una operación de E/S.
- *Terminado*. El proceso ha finalizado su ejecución.

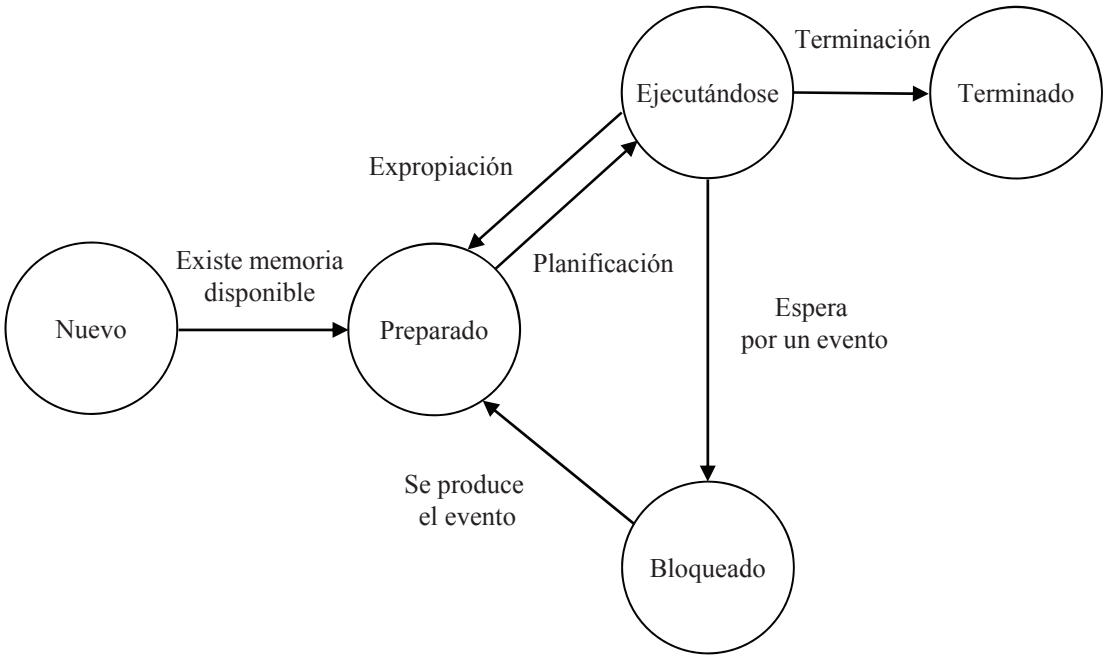

Figura 1

Solución Ejercicio 3

a) En la Figura 2a se muestra el diagrama de uso del procesador pedidos para un algoritmo de planificación de turno rotatorio con un cuanto $q = 2$ ut.

b) En Tabla 1 se muestran los tiempos de llegada T_{LL}, finalización T_F, retorno T_R, servicio T_S y espera T_E de cada proceso cuando se usa un algoritmo de planificación de turno rotatorio con un cuanto $q = 2$ ut.

c) En la Figura 2b se muestra el diagrama de uso del procesador para el algoritmo de planificación basado en prioridades de tipo expropiativo. En Tabla 2 se muestran los tiempos solicitados para este algoritmo.

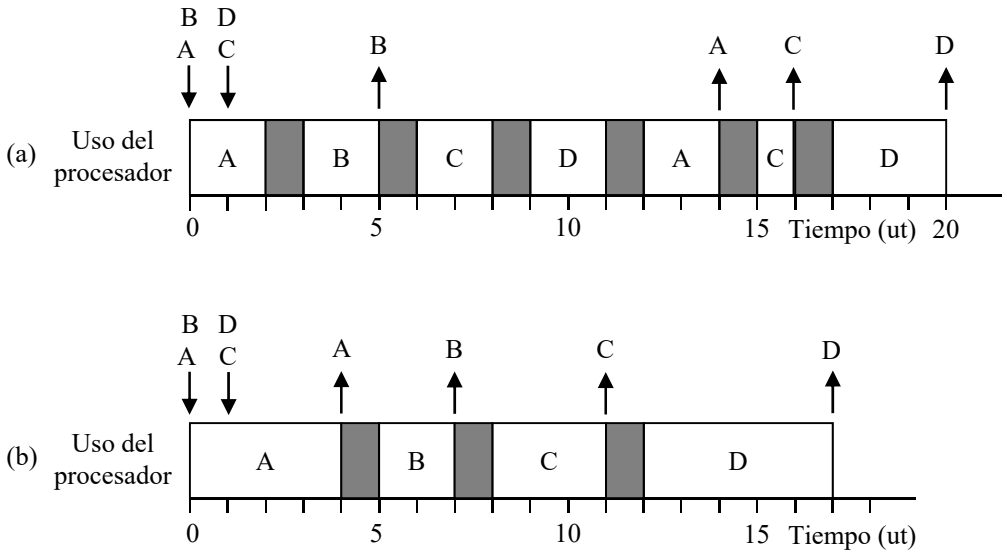

Figura 2 – Diagrama de uso del procesador usando los siguientes algoritmos de planificación: a) Turno rotatorio con $q = 2$ ut. b) Planificación por prioridades de tipo expropiativo.

Proceso	$\mathbf{T_{LL}}$ **(ut)**	$\mathbf{T_F}$ **(ut)**	$\mathbf{T_R = T_F - T_{LL}}$	$\mathbf{T_S}$ **(ut)**	$\mathbf{T_E = T_R - T_S}$
A	0	14	14	4	10
B	0	5	5	2	3
C	1	16	15	3	12
D	1	20	19	5	14

Tabla 1 – Tiempos de cada proceso con planificación de turno rotatorio con un cuanto $q = 2$ ut.

Proceso	$\mathbf{T_{LL}}$ **(ut)**	$\mathbf{T_F}$ **(ut)**	$\mathbf{T_R = T_F - T_{LL}}$	$\mathbf{T_S}$ **(ut)**	$\mathbf{T_E = T_R - T_S}$
A	0	4	4	4	0
B	0	7	7	2	5
C	1	11	10	3	7
D	1	17	16	5	11

Tabla 2 – Tiempos de cada proceso con planificación basada en prioridades de tipo expropiativa.

Solución Ejercicio 4

En la Figura 3 se propone una definición de monitor que utiliza las siguientes variables globales y variables de condición:

- `contador`. Variable global de tipo entero para llevar la cuenta del número de ciudadanos en la cola o en las ventanillas.
- `ventanilla_disponible`. Variable de condición para bloquear en su cola a los procesos hasta que exista alguna ventanilla disponible.

```
#define N 4 /* Número de ventanillas */
monitor oficina /* Definición del monitor */
    condición ventanilla_disponible;
    int contador;

    int obtener_ventanilla()  /* Procedimiento del monitor */
    {
      if (contador == N) wait_mon(ventanilla_disponible);
      contador=contador+1;
    }

    void dejar_ventanilla() /* Procedimiento del monitor */
    {
      contador = contador - 1;
      signal_mon(ventanilla_disponible);
    }

    { /* Inicialización del monitor */
      contador=0;
    }
end monitor

void ciudadano() /* Proceso ciudadano */
{
    oficina.obtener_ventanilla();
    realizar_gestión();
    oficina.dejar_ventanilla();
}

main() /* Ejecución concurrente */
{
    ejecución_concurrente(ciudadano,...,ciudadano);
}
```

Figura 3