

marinamuca01

www.wuolah.com/student/marinamuca01

14204

Ejercicios-Tema-2-SO-resueltos.pdf

Ejercicios Tema 2 SO Resueltos



2º Sistemas Operativos



Grado en Ingeniería Informática



Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación
Universidad de Granada



Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.



Exámenes, preguntas, apuntes.

12:48

WUOLAH

Join the student revolution.

MULTI

Conéctate dónde y cómo prefieras.

Guarda tus apuntes en un lugar seguro y ordenado, y accede a ellos desde tu pc, móvil o tablet.

Acceder

Registrarse

GET IT ON
Google Play

Download on the
App Store

TEMA 2

Problemas

1. Cuestiones generales sobre procesos y asignación de CPU:

- ¿Cuáles son los motivos que pueden llevar a la creación de un proceso?

¿Por qué motivos se puede crear un proceso?

- En sistemas batch → en respuesta a la recepción y admisión de un trabajo
- "Logon" interactivo → un usuario se autentica desde un terminal (log on), el SO crea un proceso que ejecuta el intérprete de órdenes asignado.
- El SO puede crear un proceso para llevar a cabo un servicio solicitado por un proceso de usuario.
- Un proceso puede crear otros procesos formando un árbol de procesos (relación padre-hijo).

- ¿Es necesario que lo último que haga todo proceso antes de finalizar sea una llamada al sistema para finalizar de forma explícita, por ejemplo exit()?

No, es necesario que realice una llamada a cualquier algoritmo de Kernel sys_exit() ↗
abort
exit
kill ...

- Cuando un proceso pasa a estado "BLOQUEADO", ¿Quién se encarga de cambiar el valor de su estado en el descriptor de proceso o PCB?

El cambio del valor de estado en el PCB se cambia en la llamada a la Rutina E/S en concreto lo realiza el dispatcher dentro del context_switch();

- ¿Qué debería hacer cualquier planificador a corto plazo cuando es invocado pero no hay ningún proceso en la cola de ejecutables?

Esta situación no se da nunca, ya que cuando no hay otros procesos se ejecuta IDLE (realiza estadísticas y cálculos, tiene la mínima prioridad).

- ¿Qué algoritmos de planificación quedan descartados para ser utilizados en sistemas de tiempo compartido?

Se descartan aquellos que puedan ocasionar inanición → Shortest Job First y prioridades
El más adecuado para tiempo compartido es Round Robin (Quantum)

2. Cuestiones sobre el modelo de procesos extendido:

- ¿Qué pasos debe llevar a cabo un SO para poder pasar un proceso de reciente creación de estado “NUEVO” a estado “LISTO”?

Para pasar de NUEVO a LISTO actúa el program loader(), éste realiza los siguientes pasos:

- Crea y reserva un PCB
 - Inicializa el PCB creado
 - Crea la región de Pila
 - Carga el programa en RAM
 - Cambia el valor del estado en el PCB a LISTO
 - Por último encola el proceso en la lista de listos.
- ¿Qué pasos debe llevar a cabo un SO para poder pasar un proceso ejecutándose en CPU a estado “FINALIZADO”?

Para pasar de EJECUTÁNDOSE a FINALIZADO se llama al algoritmo de kernel sys_exit(), éste realiza los siguientes pasos:

- Cambia el valor del estado en el PCB a FINALIZADO
 - Libera los recursos
 - Libera el PCB
 - Llama al context switch para dar paso al siguiente proceso en la cola de listos.
- Hemos explicado en clase que la función context_switch() realiza siempre dos funcionalidades y que además es necesario que el kernel la llame siempre cuando el proceso en ejecución pasa a estado “FINALIZADO” o “BLOQUEADO”. ¿Qué funcionalidades debe realizar y en qué funciones del SO se llama a esta función?

El context_switch llama al planificador de CPU que selecciona el siguiente proceso de la cola de listos y, posteriormente, llama al dispatcher que da el control de la CPU al proceso seleccionado.

Se llama en las siguientes funciones

sys_exit();
rutina_EIS();
RSI_Reloj();

- Indique el motivo de la aparición de los estados “SUSPENDIDO-BLOQUEADO” y “SUSPENDIDO-LISTO” en el modelo de procesos extendido.

En sistemas multiprogramados los estados Suspendido Listo y suspendido bloqueado surgen cuando no hay suficiente espacio en RAM o para mejorar la mezcla de procesos, entonces es necesario sacar los procesos a memoria secundaria para reducir el grado de multiprogramación, luego volverlos a introducir, es lo que se conoce como intercambio (swapping).

Los procesos pasan de LISTO a SUSP_LISTO y de BLOQUEADO a SUSP_BLOQUEADO

3. ¿Tiene sentido mantener ordenada por prioridades la cola de procesos bloqueados? Si lo tuviera, ¿en qué casos sería útil hacerlo? Piense en la cola de un planificador de E/S, por ejemplo el de HDD, y en la cola de bloqueados en espera del evento “Fin E/S HDD”.

No tendría sentido porque lo más eficiente sería que, por ejemplo en el caso de procesos que esperan por una E/S de HDD, se fuesen desbloqueando los procesos conforme finaliza la E/S por la que esperan. Si se implementase una cola con prioridad podría darse que hubiese terminado el evento E/S y no se desbloquease por tener una baja prioridad.

4. Explique las diferentes formas que tiene el kernel de ejecutarse en relación al contexto de un proceso y al modo de ejecución del procesador.

El procesador tiene dos modos de ejecución: usuario y kernel

En el modo usuario se ejecuta en el contexto de un proceso el código de usuario. No está permitido que se ejecute código en contexto de kernel en este modo.

En el modo privilegiado se ejecuta en el contexto de un proceso se ejecutan las llamadas al sistema y las excepciones. Además, en el contexto de kernel se ejecuta en este modo el tratamiento de interrupciones y las tareas del sistema.

5. Responda a las siguientes cuestiones relacionadas con el concepto de hebra:

- ¿Qué elementos de información es imprescindible que contenga una estructura de datos que permita gestionar hebras en un kernel de SO? Describa las estructuras task_t y la thread_t.

TID, contexto registros, pila, puntero a todas los recursos compartidos del kernel, estado, parámetros de planificación

TCB (thread_t) es la estructura de datos que el kernel requiere para gestionar las hebras

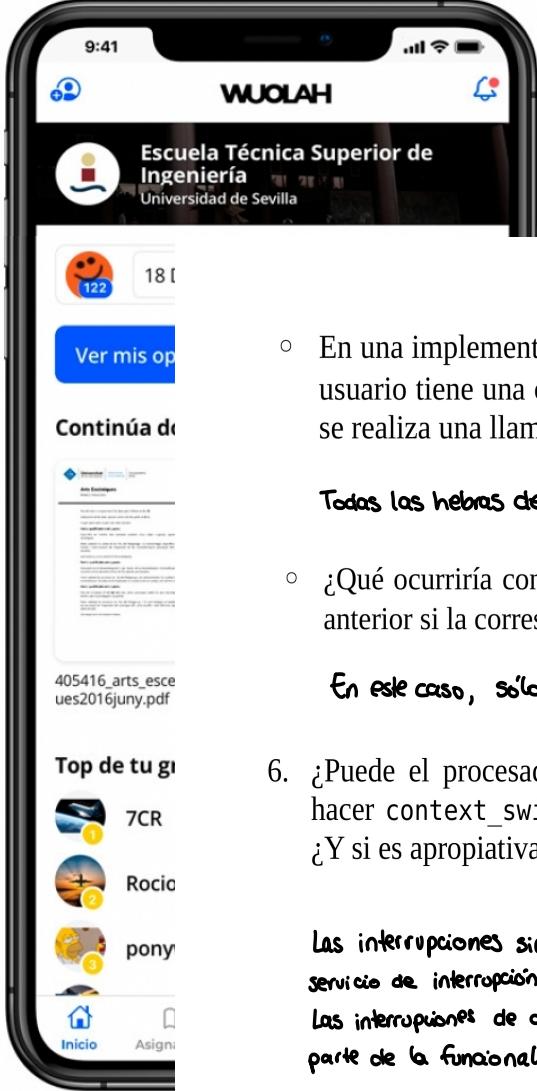
Ésta incluye el contexto de registros, la pila de ejecución y el estado. Además incluye también un TID y punteros a la lista de listos.

El PCB(task_t) ahora contiene los estados Nuevo, Finalizado, Suspendido_LISTO, Suspendido_BLOQUEADO, una lista de TCBs de las hebras que forman el proceso y el TCB de la hebra principal.

```
// PCB multithreading
PCB {
    PID (el ID_proceso);
    ESTADO = {"Nuevo", "Finalizado",
              "Suspendido_LISTO", "Suspendido_BLOQUEADO"};
    // Espacio de direcciones (TEXTO + DATOS)
    // SWAPPING
    Bloques de disco ocupados
    // Resto recursos (archivos, señales,...)
}

LISTA de Thread Control Block (TCB)
main_thread_TCB
...
}
```

```
TCB {
    TID (el ID_hebra);
    ESTADO = {"LISTO", "Ejecutándose", "Bloqueado",
              "Nuevo", "Finalizado", ...};
    CONTEXTO REGISTROS CPU {
        PC, PSW
        BP, SP //Pila de ejecución
    }
    LISTA_LISTOS *next; LISTA_LISTOS *previous;
}
```



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

[Available on the App Store](#) [GET IT ON Google Play](#)

- En una implementación de hebras con una biblioteca de usuario en la cual cada hebra de usuario tiene una correspondencia N:1 con una hebra kernel, ¿Qué ocurre con la tarea si se realiza una llamada al sistema bloqueante, por ejemplo read()?

Todas las hebras de la tarea se bloquean también.

- ¿Qué ocurriría con la llamada al sistema read() con respecto a la tarea de la pregunta anterior si la correspondencia entre hebras usuario y hebras kernel fuese 1:1?

En este caso, sólo se bloquearía la hebra que realizase la tarea bloqueante.

6. ¿Puede el procesador manejar una interrupción mientras está ejecutando un proceso sin hacer context_switch() si la política de planificación que utilizamos es no apropiativa? ¿Y si es apropiativa?

Las interrupciones siempre se pueden procesar sin hacer cambio de contexto, ya que las RSI (rutas de servicio de interrupción) se ejecutan en el contexto del proceso que está actualmente ejecutándose en la CPU. Las interrupciones de dispositivos y sus correspondientes RSIs son para control de la EIS y forman parte de la funcionalidad de SO asociada a control del sistema. Siempre se manejan.

Políticas apropiativas:

```
RSI() {
    ...
    if(Planif_CPU (PID_VaEntrarAListos, PID_CPU) == true)
        dispatch(PIC_CPU, PID_VaEntrarAListos);
    else{
        PID_ESTADO = "LISTO";
        encolar(PID.LISTA_LISTOS);
    }
}
```

Políticas no apropiativas:

```
RSI {
    // OK o error
    // Transferencia de info. ...
    // ... ModuloE/S-CPU-RAM
    PID.ESTADO="LISTO";
    encolar(PID,LISTA_LISTOS);
    iret;
}
```

7. Suponga que es responsable de diseñar e implementar un SO que va a utilizar una política de planificación apropiativa (*preemptive*). Suponiendo que el sistema ya funciona perfectamente con multiprogramación pura y que tenemos implementada la función Planif_CPU(), ¿qué otras partes del SO habría que modificar para implementar tal sistema? Escriba el código que habría que incorporar a dichas partes para implementar apropiación (*preemption*).

Habrá que modificar RSI(), program_loader() y Planif_Medio_Plano() añadiéndole:

```
if(Planif_CPU (PID_VaEntrarAListos, PID_CPU) == true)
    dispatch(PIC_CPU, PID_VaEntrarAListos);
else{
    PID_ESTADO = "LISTO";
    encolar(PID.LISTA_LISTOS);
}
```

8. Para cada una de las siguientes llamadas al sistema explique si su procesamiento por parte del SO requiere la invocación del planificador a corto plazo (Planif_CPU()):
- Crear un proceso, fork().

Cuando se crea un proceso pasa de estado NUEVO a LISTO, éste cambio lo realiza el program-loader(). Por tanto, sólo se invoca al Planif_CPU() si el sistema es apropiativo.

- Abortar un proceso, es decir, terminarlo forzosamente, abort().

Siempre porque para pasar de Ejecutándose a Finalizado (sys_exit) se llama al context switch y éste llama al Planif_CPU, para que un proceso nuevo disfrute de la CPU.

- Bloquear (suspender) un proceso, read() o wait().

Pasa de EJECUTÁNDOSE a BLOQUEADO (rutina E/S), en el código de la rutina E/S se llama a context_switch() y éste al Planif_CPU().

- Desbloquear (reanudar) un proceso, RSI o exit() (complementarias a las del caso anterior).

Cuando se crea un proceso pasa de estado BLOQUEADO a LISTO, éste cambio lo realiza la RSI(). Por tanto, sólo se invoca al Planif_CPU() si el sistema es apropiativo.

- Modificar la prioridad de un proceso.

Si al modificarla llega a LISTO y tiene más prioridad que el primero de la cola sí

9. En el algoritmo de planificación FCFS, el índice de penalización, $(M+r)/r$, ¿es creciente, decreciente o constante respecto a r (ráfaga de CPU: tiempo de servicio de CPU requerido por un proceso)? Justifique su respuesta.

En FCFS todos los procesos pierden la misma cantidad de tiempo esperando en la cola de ejecutables independientemente de sus necesidades. $\Rightarrow M$ es constante.

Si M es constante, a mayor r , menor índice de penalización \Rightarrow es decreciente.

Procesos cortos muy penalizados y largos poco penalizados.

10. Sea un sistema multiprogramado que utiliza el algoritmo Por Turnos (Round-Robin, RR). Sea S el tiempo que tarda el despachador en cada cambio de contexto. ¿Cuál debe ser el valor de quantum Q para que el porcentaje de uso de la CPU por los procesos de usuario sea del 80%?

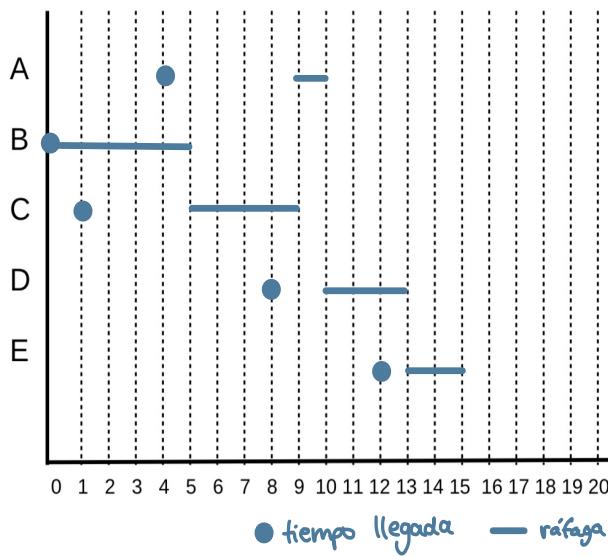
$$\begin{aligned} r &\rightarrow 80\% \quad | \text{ tiempo de CPU (Quantum)} \\ S &\rightarrow 20\% \end{aligned}$$

$$20\%Q = S \Rightarrow Q = 0.2S$$

11. Para la siguiente tabla que especifica una determinada configuración de procesos, tiempos de llegada a cola de listos y ráfagas de CPU; responda a las siguientes preguntas y analice los resultados:

Proceso	Tiempo de llegada	Ráfaga CPU
A	4	1
B	0	5
C	1	4
D	8	3
E	12	2

- FCFS. Tiempo medio de respuesta, tiempo medio de espera y penalización.



• Tiempo medio espera (\bar{M}_w)
↳ separación entre llegada y ráfaga

$$\left. \begin{array}{l} M_A = 5 \\ M_B = 0 \\ M_C = 4 \\ M_D = 2 \\ M_E = 1 \end{array} \right\} \bar{M}_w = (M_A + M_B + M_C + M_D + M_E) / 5 \quad \bar{M}_w = 2^1.4$$

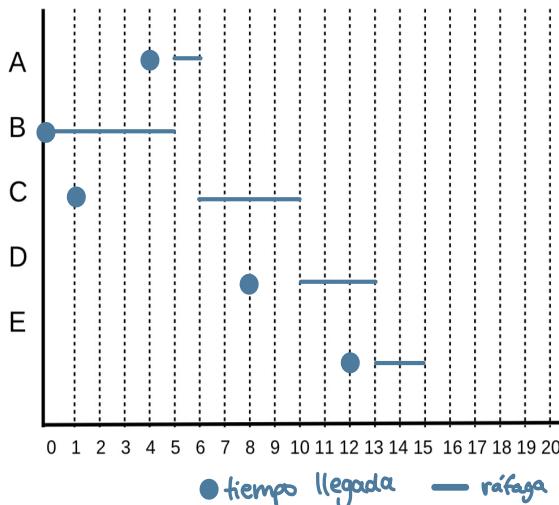
• Tiempo medio de respuesta (\bar{T}) $\rightarrow T = M + r$

$$\left. \begin{array}{ll} T_A = 6 & T_0 = 5 \\ T_B = 5 & T_E = 3 \\ T_C = 8 & \end{array} \right\} \bar{T} = 27 / 5 = 5^1.4$$

• Tiempo medio penalización (\bar{P}) $\rightarrow P = T/r$

$$\left. \begin{array}{ll} P_A = 6 & P_0 = 1^1.667 \\ P_B = 1 & P_E = 1^1.5 \\ P_C = 2 & \end{array} \right\} \bar{P} = 12^1.167 / 5 = 2^1.4334$$

- SJF (ráfaga estimada coincide con ráfaga real). Tiempo medio de respuesta, tiempo medio de espera y penalización.



• Tiempo medio espera (\bar{M}_w)
↳ separación entre llegada y ráfaga

$$\left. \begin{array}{l} M_A = 1 \\ M_B = 0 \\ M_C = 5 \\ M_D = 2 \\ M_E = 1 \end{array} \right\} \bar{M}_w = (M_A + M_B + M_C + M_D + M_E) / 5 \quad \bar{M}_w = 1^1.8$$

- Tiempo medio de respuesta (\bar{T}) $\rightarrow T = M + r$

$$\begin{array}{ll} T_A = 2 & T_O = 5 \\ T_B = 5 & T_E = 3 \\ T_C = 9 \end{array}$$

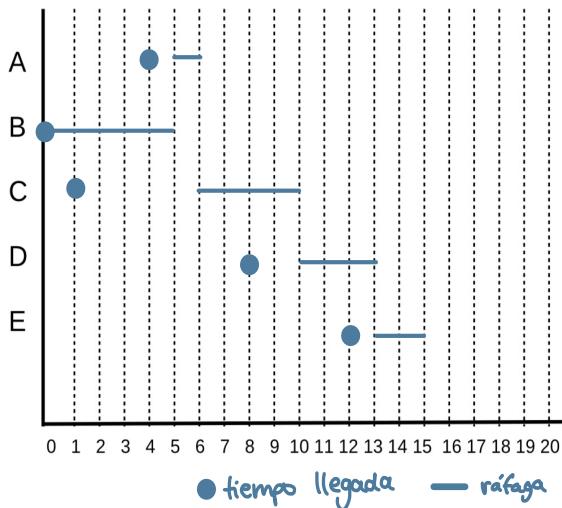
$$\bar{T} = 24 / 5 = 4'8$$

- Tiempo medio penalización (\bar{P}) $\rightarrow P = T/r$

$$\begin{array}{ll} P_A = 2 & P_O = 1'667 \\ P_B = 1 & P_E = 1'S \\ P_C = 2'25 \end{array}$$

$$\bar{P} = 8'417 / 5 = 1'6834$$

- SRTF (ráfaga estimada coincide con ráfaga real). Tiempo medio de respuesta, tiempo medio de espera y penalización.



- Tiempo medio espera (\bar{M})
 \hookrightarrow separación entre llegada y ráfaga

$$\left. \begin{array}{l} M_A = 1 \\ M_B = 0 \\ M_C = 5 \\ M_D = 2 \\ M_E = 1 \end{array} \right\} \bar{M} = (M_A + M_B + M_C + M_D + M_E) / 5$$

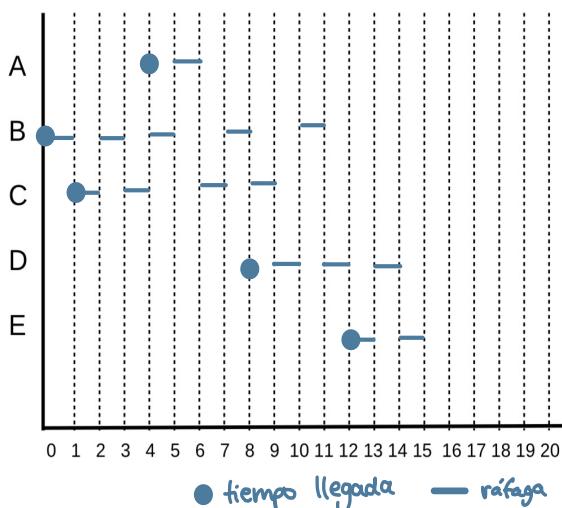
$$\bar{M} = 1'8$$

- Tiempo medio de respuesta (\bar{T}) $\rightarrow T = M + r$

$$\begin{array}{ll} P_A = 2 & P_O = 1'667 \\ P_B = 1 & P_E = 1'S \\ P_C = 2'25 \end{array}$$

$$\bar{P} = 8'417 / 5 = 1'6834$$

- RR ($q=1$). Tiempo medio de respuesta, tiempo medio de espera y penalización.



- Tiempo medio espera (\bar{M})
 \hookrightarrow separación entre llegada y ráfaga

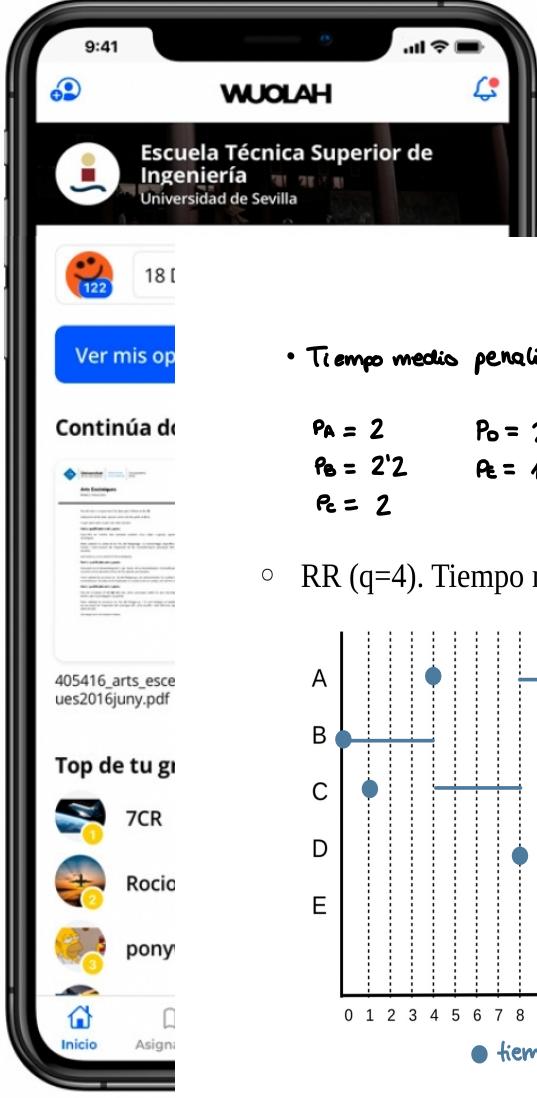
$$\left. \begin{array}{l} M_A = 1 \\ M_B = 6 \\ M_C = 4 \\ M_D = 3 \\ M_E = 1 \end{array} \right\} \bar{M} = (M_A + M_B + M_C + M_D + M_E) / 5$$

$$\bar{M} = 2'8$$

- Tiempo medio de respuesta (\bar{T}) $\rightarrow T = M + r$

$$\begin{array}{ll} T_A = 2 & T_O = 6 \\ T_B = 11 & T_E = 3 \\ T_C = 8 \end{array}$$

$$\bar{T} = 30 / 5 = 6$$



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

Escuela Técnica Superior de
Ingeniería
Universidad de Sevilla



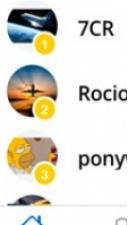
18!

Ver mis op

Continúa d



Top de tu gi



Inicio

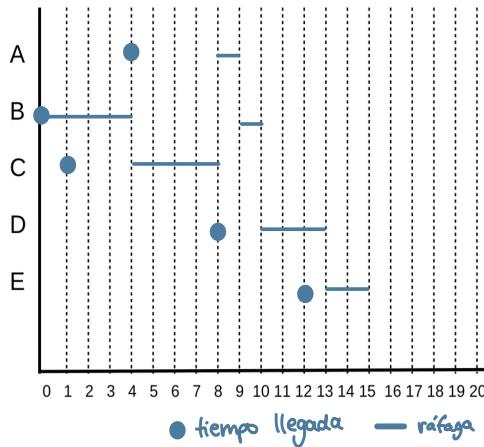
Asigni

- Tiempo medio penalización (\bar{P}) $\rightarrow P = T/r$

$$\begin{aligned} P_A &= 2 & P_0 &= 2 \\ P_B &= 2'2 & P_E &= 1'S \\ P_C &= 2 \end{aligned}$$

$$\bar{P} = 9'7 / 5 = 1'94$$

- RR (q=4). Tiempo medio de respuesta, tiempo medio de espera y penalización.



- Tiempo medio espera (\bar{M}_w)
 \hookrightarrow separación entre llegada y ráfaga

$$\left. \begin{aligned} M_A &= 4 \\ M_B &= 5 \\ M_C &= 3 \\ M_D &= 2 \\ M_E &= 1 \end{aligned} \right\} \bar{M}_w = (M_A + M_B + M_C + M_D + M_E) / 5$$

$$\bar{M}_w = 15 / 5 = 3$$

- Tiempo medio de respuesta (\bar{T}) $\rightarrow T = M + r$

$$\begin{aligned} T_A &= 5 & T_0 &= 5 \\ T_B &= 10 & T_E &= 5 \\ T_C &= 7 & T_E &= 5 \end{aligned}$$

$$\bar{T} = 30 / 5 = 6$$

- Tiempo medio penalización (\bar{P}) $\rightarrow P = T/r$

$$\begin{aligned} P_A &= 5 & P_0 &= 1'67 \\ P_B &= 2 & P_E &= 1'S \\ P_C &= 1'75 \end{aligned}$$

$$\bar{P} = 11'92 / 5 = 2'384$$

12. Utilizando los datos de la tabla del ejercicio anterior dibuje el diagrama de ocupación de CPU para el caso de un sistema que utiliza un algoritmo de colas múltiples con realimentación con las siguientes colas:

Cola	Prioridad	Quantum
1	1	1
2	2	2
3	3	4

Tenga en cuenta las siguientes suposiciones:

- Todos los procesos inicialmente entran en la cola de mayor prioridad (menor valor numérico).
- Cada cola se gestiona mediante la política RR y la política de planificación entre colas es por prioridades no apropiativa.
- Un proceso en la cola i pasa a la cola i+1 si consume un quantum completo sin bloquearse.
- Cuando un proceso llega a la cola de menor prioridad, permanece en ella hasta que finaliza.

