Planificador Rate Monotonic RM

- Todas las tareas son periódicas e independientes unas de otras
- El plazo de finalización de cada tarea se tomará igual a su periodo
- La asignación de prioridades se hará de forma inversa al periodo
- El planificador será expulsivo
- El tiempo de cambio de contexto se considera despreciable
- El conjunto de tareas es síncrono

RM óptimo en tareas periódicas simples

- **Definición**: Un conjunto de N tareas es periódico simple cuando para cualquier par de tareas T_i, T_k con $P_i < P_k$, entonces P_k es un número entero de veces P_i
- **Teorema**: Un sistema de tareas periódico simple, independientes e interrumpibles, cuyos plazos de finalización son mayores o iguales que sus periodos, es planificable en un procesador de acuerdo con el algoritmo RM, si y solo si el factor de utilización total es menor o igual que uno

Test de garantía RM
(limites de utilización)

Con las restricciones antes comentadas

- **Teorema**: Un conjunto de "n" tareas será planificable bajo el Rate-Monotonic si se cumple la siguiente desigualdad:

$$U = \sum_{i=1}^{N} \frac{Ci}{Ti} < N\left(2^{1/N} - 1\right)$$

$$\frac{C_1}{P_1} + \cdots + \frac{C_n}{P_n} = U \leq n(2^{\frac{1}{n}} - 1)$$

n=1,	1.000
n=2,	0.828
n=3,	0.779

n=4,	0.756
n=5,	0.743
n=6,	0.734

$$\lim_{n\to\infty} n(2^{\frac{1}{n}} - 1) \approx 0.693$$

Test de garantía RM (ejemplo)

	Pi	Ci
Task 1	4	1
Task 2	8	2
Task 3	12	3

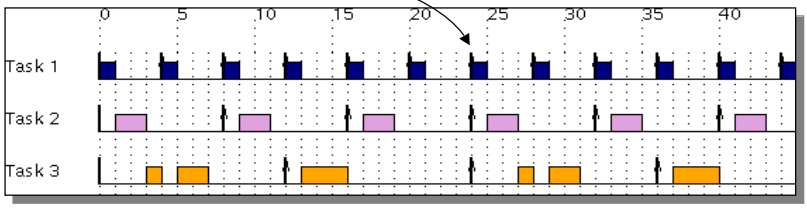
Factor de utilización

$$\frac{1}{4} + \frac{2}{8} + \frac{3}{12} = 0.75$$

Limite de Utilización

$$3(2^{\frac{1}{3}} - 1) = 0.779$$

$$mcm(4,8,12) = 24$$

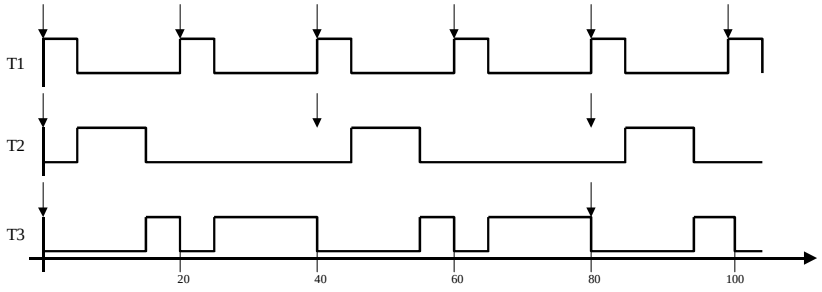


Test de garantía RM (contraejemplo)

Conjunto de tareas planificable que no cumple el test RM

Tareas	P	C	Pr	U
T1	20	5	1	0,250
T2	40	10	2	0,250
T3	80	40	3	0,500

Conjunto de tareas planificable con U=1



Test de garantía RM-DM
(tiempos de respuesta)

- **Teorema**: Un conjunto de "n" tareas será planificable bajo cualquier asignación de prioridades, si y sólo si:

$$\forall 1 \le i \le n,\ R_i \le P_i$$

$$R_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{P_j} \right\rceil C_j$$

Solución iterativa

$$w_i^0 = C_i$$

$$w_i^1 = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i^0}{P_j} \right\rceil C_j$$

$$w_i^2 = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i^1}{P_j} \right\rceil C_j$$

....

$$w_i^k = w_i^{k+1}$$

$hp(i)$ Conjunto de tareas más prioritarias que "i"

Test de garantía RM-DM (ejemplo)

Tareas	P	C	Pr	U
T1	7	3	1	0,429
T2	12	3	2	0,250
T3	20	5	3	0,250

Tabla de tareas U=0.929

$$\omega_2^1 = 3 + \left\lceil \frac{3}{7} \right\rceil 3 = 6 = \omega_2^2$$

$$\omega_3^1 = 5 + \left\lceil \frac{5}{7} \right\rceil 3 + \left\lceil \frac{5}{12} \right\rceil 3 = 11$$

$$\omega_3^2 = 5 + \left\lceil \frac{11}{7} \right\rceil 3 + \left\lceil \frac{11}{12} \right\rceil 3 = 14$$

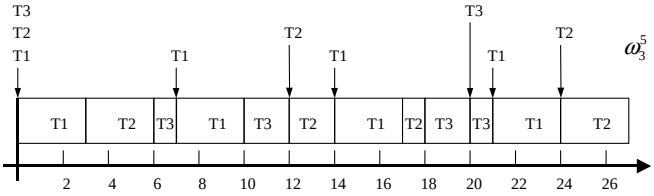
$$\omega_3^3 = 5 + \left\lceil \frac{14}{7} \right\rceil 3 + \left\lceil \frac{14}{12} \right\rceil 3 = 17$$

$$\omega_3^4 = 5 + \left\lceil \frac{17}{7} \right\rceil 3 + \left\lceil \frac{17}{12} \right\rceil 3 = 20$$

$$\omega_3^5 = 5 + \left\lceil \frac{20}{7} \right\rceil 3 + \left\lceil \frac{20}{12} \right\rceil 3 = 20$$

Tareas	T	R
T1	7	3
T2	12	6
T3	20	20

Plazos de finalización



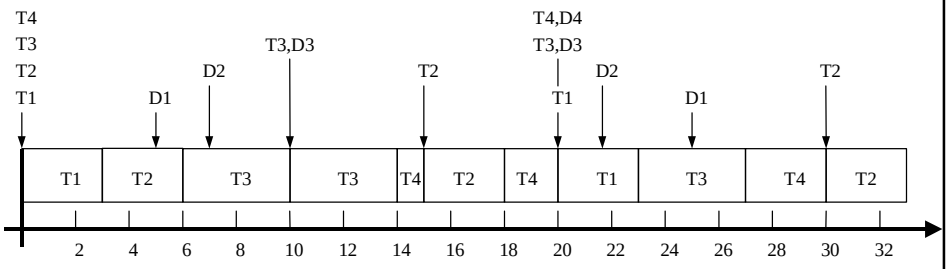
Planificador Deadline Monotonic DM

- Es una política de planificación similar a RM
- Se elimina la restricción $D_i = P_i$
- Ahora los plazos de finalización pueden ser inferiores a los periodos: $D_i \leq P_i$
- En este caso, la asignación de prioridades se hace en orden inverso al plazo de finalización
- Como test de garantía se utiliza el análisis de los tiempos de respuesta (test RM – DM), pero verificando ahora que $R_i \leq D_i$

Test de garantía DM (ejemplo)

Tareas	P	D	C	Pr	R
T1	20	5	3	1	3
T2	15	7	3	2	6
T3	10	10	4	3	10
T4	20	20	3	4	20

Tabla de tareas. U=0.90



DM óptimo en tareas de prioridades fijas

- DM no es óptimo en el caso más general, pero sí cuando las prioridades son fijas
- **Teorema**: Un sistema de tareas independientes e interrumpibles, que están en fase y sus plazos de finalización son menores o iguales que sus respectivos periodos, pueden ser planificadas en un único procesador bajo la política DM sí y solo sí pueden ser planificadas con cualquier otro planificador de prioridades fijas
- **Corolario**: El algoritmo de planificación RM es óptimo entre los algoritmos de planificación de prioridades fijas cuando los plazos de finalización de las tareas son proporcionales a sus periodos ($D_i = \delta P_i$, $\delta > 0$, $\forall$ i)

Tiempo de ejecución en el peor caso

- Hasta ahora hemos asumido que se conoce C_i
- Se puede obtener una aproximación utilizando métodos de medida y análisis
- Actividades en los métodos de análisis:
  - Descomponer el código de la tarea en bloques básicos, construyendo un grafo de ejecución
  - Estimar el tiempo de ejecución de cada bloque básico a partir del modelo de procesador
  - A partir de los tiempos de ejecución se va simplificando el grafo de ejecución
- El tiempo de ejecución debe ser determinista (bucles y llamadas recursivas acotadas)
- Además del procesador, se debe estudiar el código generado por el compilador (optimizaciones)

Mejora del modelo simple

- Las restricciones del modelo simple nos han facilitado obtener el test de garantía en las políticas RM y DM
- Ahora vamos a eliminar las siguientes restricciones
  - El tiempo de cambio de contexto es nulo
  - Las tareas son independientes
  - Todas las tareas son periódicas
- Y estudiaremos como afectan estos cambios al planificador y al test de garantía

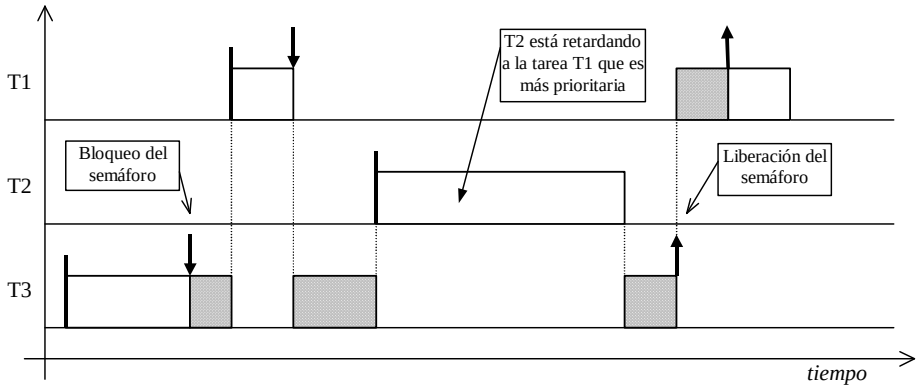
Cambio de contexto

- El tiempo de cambio de contexto C_s consiste en tres operaciones:
  - Decidir la siguiente tarea a ejecutar
  - Salvar el estado del proceso en ejecución
  - Restaurar el estado del nuevo proceso
- Cada expulsión de una tarea por la llegada de otra prioritaria implica dos cambios de contexto:
  - El primero retrasa la entrada de la mas prioritaria
  - El segundo retrasa la reincorporación de la menos prioritaria
- Nosotros añadiremos estos dos tiempos a la tarea más prioritaria (resulta más fácil de contabilizar)
- En el test de garantía hay que sustituir C_i por $C_i' = C_i + 2C_s$, en todas las tareas excepto en la menos prioritaria

Gestión de recursos

- Queremos eliminar la restricción de que las tareas son independientes. Esto engloba:
  - La sincronización entre tareas
  - La comunicación
  - La utilización de recursos comunes
- Solo contemplaremos el uso de semáforos para resolver el uso de secciones críticas
- En los sistemas de tiempo real es importante evitar el fenómeno de la inversión de prioridad. No es admisible pues convierte al sistema en indeterminista

Ejemplo de inversión de prioridad



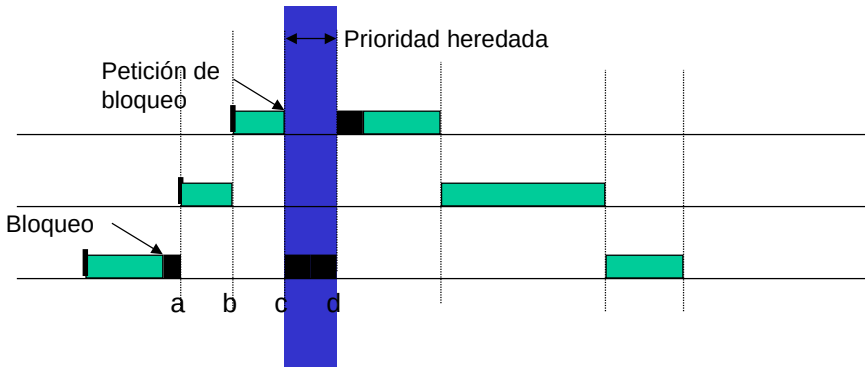
Mecanismos para evitar la Inv. Prioridad

- Ordenar la ejecución de las tareas modificando las prioridades. Puede hacer al sistema no planificable
- Con secciones críticas no interrumplibles
- Con secciones críticas interrumpibles, modificando las prioridades de las tareas que entran en las secciones críticas. Son protocolos de herencia de prioridad:
  - Priority Inheritence Protocol
  - Priority Ceiling Protocol
  - Priority Semafore Protocol
- Estos últimos utilizan semáforos para acceder a las regiones críticas, con unas reglas que afectan a la prioridad de las tareas
- Estudiaremos como afectan al test de planificabilidad

Proprity Inheritance Protocol

- Funcionamiento:
  - Cuando una tarea T_h intenta acceder a una sección crítica que esté bloqueada por otra tarea T_l, la tarea bloqueante T_l hereda la prioridad de la tarea más prioritaria que quiere acceder T_h
  - La herencia de prioridad solo se produce cuando una tarea de más prioridad queda bloqueada. No se produce si la tarea T_l que queda bloqueada tiene menor prioridad que la propia T_h
  - Al liberar el semáforo de la sección crítica, la tarea T_l recupera su prioridad propia
- Se debe tener en cuenta el caso en que la tarea T_l esté, a su vez, bloqueada en otra sección crítica

PIP. Ejemplo



PIP. Propiedades

- Una tarea de alta prioridad sólo puede ser bloqueada por una de baja prioridad si esta última está dentro de una zona crítica cuando se activa la primera
- Una tarea de menos prioridad puede bloquear a otra de más prioridad, como mucho durante una zona crítica
- La sección crítica de mayor duración determina el tiempo máximo que una tarea puede bloquear a otra de mayor prioridad
- El número de veces que una tarea puede quedar bloqueada viene dado por el min(n, m), donde “n” es el número de tareas de menor prioridad y “m” el número de regiones críticas usados por estas

PIP. Inconvenientes

- No evita el interbloqueo
- El tiempo de bloqueo puede ser excesivo

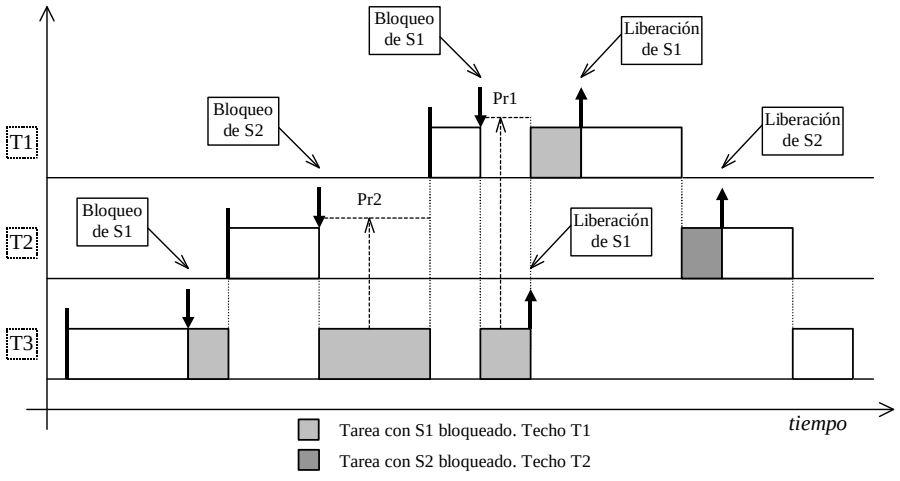
Test de garantía

- Hay que tener en cuenta el tiempo máximo que se puede retrasar una tarea
- Se tendrá en cuenta este retraso en el cálculo del tiempo de respuesta

Proirity Ceiling Protocol

- Funcionamiento
  - A cada semáforo se le asigna una prioridad igual a la de la tarea más prioritaria que le puede bloquear
  - Una tarea puede cerrar un semáforo si su prioridad es estrictamente mayor que la prioridad de todos los semáforos que en este momento estén cerrados (no se le sube la prioridad)
  - Una tarea mantiene su prioridad mientras no bloquee a otras tareas más prioritarias, en cuyo caso hereda la prioridad máxima de entre las que bloquea
  - Al abandonar la tarea una sección crítica recupera su prioridad

PCP. Ejemplo



Propiedades

- Impide el interbloqueo, aunque se puede evitar con otros algoritmos
- No existen bloqueos encadenados
- El máximo tiempo de espera está acotado por la sección crítica más larga de las tareas menos prioritarias

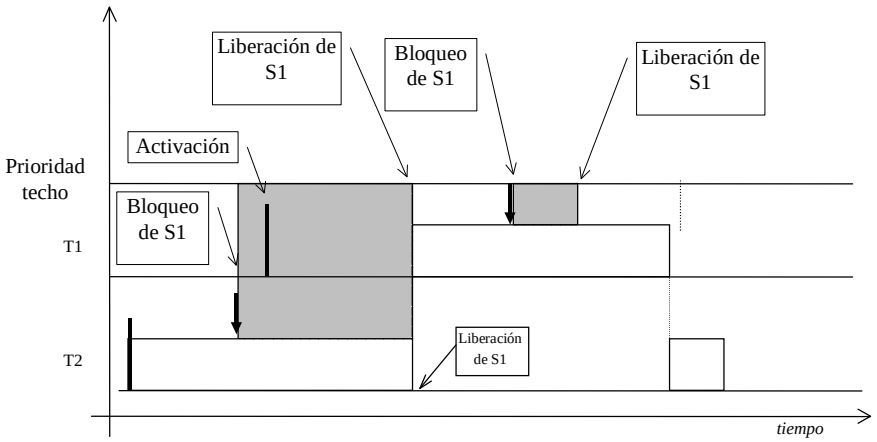
Problemas

- Es un algoritmo pesimista, es decir, puede bloquear tareas intermedias sin necesidad
- Es complejo de implementar. Por este motivo no es muy utilizado

Prority Semafore Protocol

- Funcionamiento
  - A cada semáforo se le asigna una prioridad igual a la de la tarea más prioritaria que le puede bloquear
  - Cuando una tarea entra en la sección crítica, hereda la prioridad que se le ha asignado a su semáforo
- El funcionamiento es similar a hacer las secciones críticas no interrumpibles, excepto para las tareas más prioritarias que no utilizan recursos
- El problema que presenta es que interfiere más al funcionamiento de otras tareas que el PCP
- El máximo tiempo de espera de una tarea está acotado también por la sección crítica más larga de las tareas de menor prioridad

PSP. Ejemplo



Test de garantía

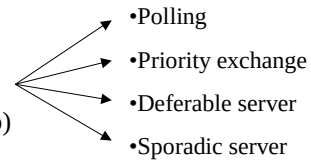
- El test de planificabilidad se modifica añadiendo el factor de interferencia B_i que corresponderá al tiempo máximo que cada tarea puede estar bloqueada por otras tareas que utilizan sus mismos recursos

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j + B_i$$

- El factor de interferencia dependerá del protocolo que se esté utilizando para el uso de los semáforos
- Por ejemplo, en el PCP corresponde a la sección crítica más larga de las tareas menos prioritarias
- El test deja de ser una condición necesaria

Servicio aperiódico

- En un STR es usual la existencia de tareas sin plazos de finalización (aperiódicas)
- Una posibilidad es ejecutarlas en el tiempo sobrante, pero puede ser deseable un tiempo de respuesta mejor
- El servicio que se encarga de ejecutar las tareas aperiódicas se llama servicio aperiódico. Tiene la forma de una tarea (periódica) integrada en el sistema que ejecuta el trabajo aperiódico según se va activando
- Tipos de servidores:
  - Background
  - Bandwith preserving
  - Slack Stealing (óptimo)
    - Estático
    - Dinámico



Servidor Background

- Cuando no hay trabajo periódico que ejecutar se sirven las tareas aperiódicas
- Se implementa con una tarea periódica con tiempo de computación y periodos grandes, dándole la prioridad más baja del sistema
- Se le llama servidor porque la tarea va ejecutando los cuerpos de las distintas activaciones de las tareas aperiódicas.
- Cuando se activan las tarea aperiódicas se ponen en una cola de la cual se extraen por el servidor aperiódico en orden de llegada

Servidor Polling

- Se añade una tarea periódica a las tareas críticas
- Durante el tiempo de cómputo del servidor se ejecutará el trabajo aperiódico, si lo hay
- Para que el sistema siga siendo planificable se le asigna a la tarea un P_i y C_i adecuado. La prioridad será la que le corresponda según el planificador
- El tiempo de cómputo será la capacidad del servidor para ejecutar trabajo aperiódico. Si un trabajo aperiódico necesita más del C_i se servirá en varias activaciones del servidor
- El tiempo de respuesta es mejor que el Background pues el servidor no tiene la prioridad más baja
- Tiene la desventaja de que si está activo el servidor y no hay trabajo aperiódico debe seguir consumiendo tiempo de CPU

Servidor Priority Excange

- Pretende eliminar el inconveniente del servidor anterior, para no perder el tiempo cuando no hay trabajo aperiódico
- También es una tarea periódica de prioridad intermedia con las siguientes consideraciones:
  - Si cuando el servidor está activo hay tareas aperiódicas pendientes, estas se sirven hasta que se consume todo el tiempo del servidor
  - Si no hay trabajo que servir, intercambia su prioridad con la tarea periódica más prioritaria que esté lista en ese momento (de las menos prioritarias que el), y sólo durante el tiempo de cómputo del servidor
  - La capacidad del servidor se repone completamente al inicio de cada periodo (la prioridad del servidor va disminuyendo y se va gastando su tiempo de cómputo)
- Tiene el inconveniente de que hay que modificar el planificador para que soporte el intercambio de prioridad

Servidor Deferable Server

- Añade una tarea periódica de prioridad intermedia con las siguientes consideraciones:
  - Si cuando el servidor está activo hay tareas aperiódicas pendientes, estas se sirven hasta que se consume todo el tiempo del servidor. Para ello se está llevando la cuenta de su C_i hasta que se agota
  - La capacidad del servidor se repone completamente al inicio de cada periodo
- La capacidad que puede soportar es peor que el anterior pues se incumple la condición de que una tarea no se puede suspender a si misma. La situación de carga podría variar a puntos que no corresponden al comienzo del periodo.
- El tiempo de cómputo que ofrece es bajo

Servidor Sporadic Server

- Se añade una tarea periódica con las siguientes características:
  - Conserva su capacidad cuando no hay trabajo aperiódico al igual que el Deferable Server
  - El mecanismo de relleno viene dado por el instante en que se hace y la cantidad de tiempo que se rellena
    - **Instante** de relleno: Cuando el servidor se activa, y su capacidad es mayor que cero, se programa un instante de relleno igual al instante actual más el periodo del servidor
    - **Cantidad** a rellenar: Se determina cuando el servidor pasa de activo a ocioso, o cuando se ha agotado la capacidad del servidor. En este momento se decide que la cantidad a rellenar es igual a la cantidad consumida desde la última vez que pasó de ocioso a activo
  - El servidor se planifica como una tarea periódica normal
- Desaparece el inconveniente del anterior pues la capacidad consumida se recupera a partir del periodo

Servidor Slack Stealing

- El mecanismo del servidor consiste en averiguar cuando es lo más tarde que se puede ejecutar el trabajo periódico, de manera que se sigan respetando los plazos
- Al retrasar el trabajo periódico se adelanta la ejecución del trabajo aperiódico
- No se añade una tarea nueva. Se ejecuta el trabajo aperiódico en los tiempos de cómputo que han quedado libres
- En el **planificador estático** se calculan los tiempos de activación de todas las tareas hasta el hiperperiodo. Cuando se activa una tarea aperiódica se decide si se va a ejecutar consultando la tabla de activaciones
- Lo complicado es adelantar la ejecución de las tareas cuando no hay trabajo aperiódico
- En el **planificador dinámico** se intenta evitar la construcción de la tabla. Se basa en calcular en el momento que llega trabajo aperiódico el trozo de tabla que afecta en ese momento y decidir si se va a ejecutar
- El planificador dinámico es óptimo, pero los tiempos de cálculo son elevados

ERROR: syntaxerror
OFFENDING COMMAND: --nostringval--

STACK:

()
/Title
()
/Subject
(05.04.2005 23:26:38)
/ModDate
()
/Keywords
(PDFCreator Version 0.8.1)
/Creator
(05.04.2005 23:26:38)
/CreationDate
(Gomis)
/Author
-mark-