/Zöbel95/:

You remember?

Übersicht

Wozu Prozesse?
Prozesse vs. Programme
Prozesszustände und Übergänge
- Zustandsübergänge
- PCB (Process Control Block)
- Prozessumschaltung
- Zustandsübergänge im Einzelnen (Ursachen und Aktionen)

Scheduling
- Ziele und Bedingungen
- Non-preemptive Scheduling
  - FCFS (First Come First Served)
  - Suchen
  - SJF (Shortest Job First)
  - EDF (Earliest Deadline First)
  - LLF (Least Laxity First)
  - Priority
- Preemptive Scheduling
  - Round Robin
  - SJF
  - EDF
  - Priority
- Scheduling-Kombinationen
  - Feedback Scheduling
  - Multiple Queues

Prozesse vs. Threads

Betriebssysteme – **Prozess- und Prozessorverwaltung**

Arnulf Deinzer, FH Kempten, Sommersemester 2003
3.1

BSS1 3.1

/Zöbel95/:

2 Basics of RT Planning

2.1 Process Model

2.2 Planning by Searching

2.3 EDF (Earliest Deadline First)

2.4 LLF (Least Laxity First)

2.5 Planning by Monotone Rates

2.6 Evaluation of Planning Methods

Process

First discussions only for

•one processor systems

RT scheduling: distribution of processor time to processes so that these don´t violate their deadlines.

Processes (or the OS) may be

•non preemptive

•preemptive

The switch between different processes

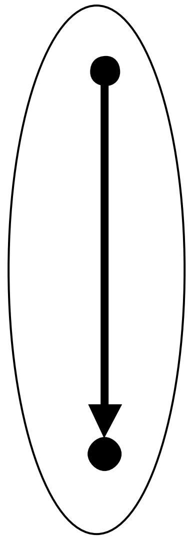
•context switch

needs time, but is not part of following discussions.

Processes may be

•periodical

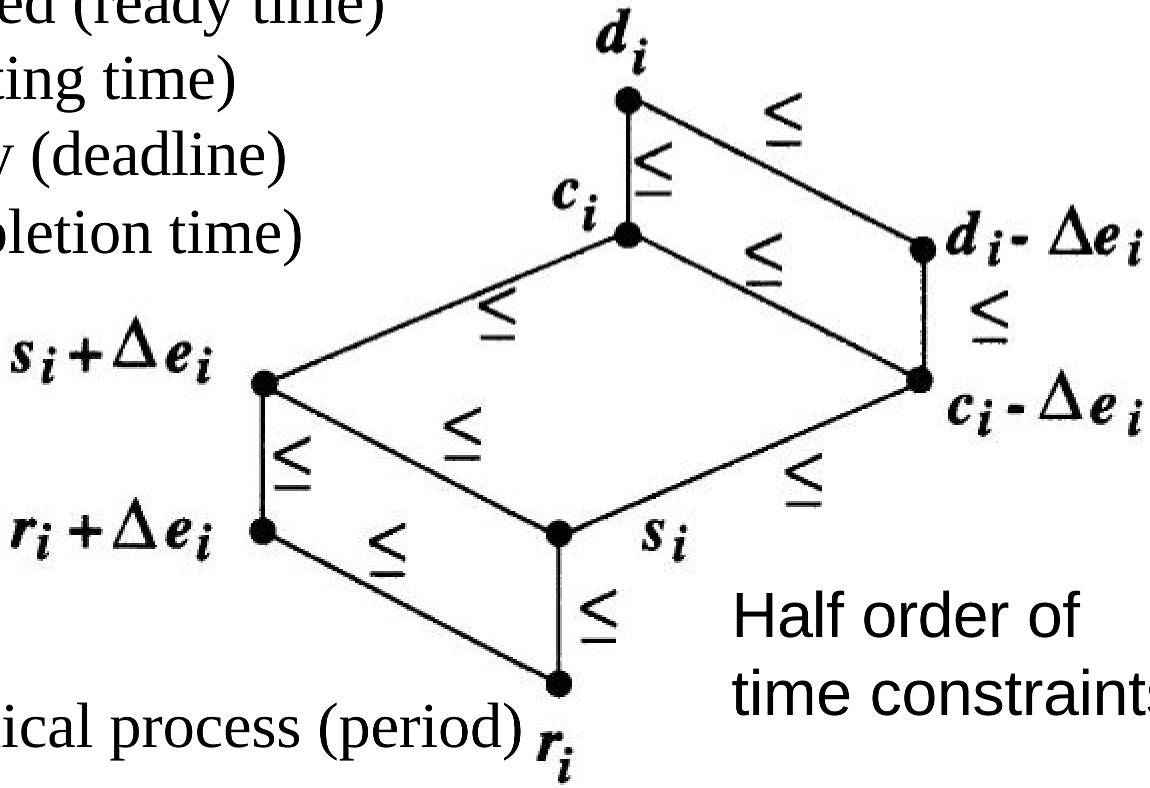
•non periodical, sporadic.



Process as sequential and limited number of statements

Process Parameters

P	Process type
P_i	Process incarnation i of a process
P_i^j	j-th execution of process incarnation i of a process
? e_i	the time P_i needs for its execution (execution time)
r_i	be the time P_i can be started (ready time)
s_i	the time P_i is started (starting time)
d_i	the time P_i has to be ready (deadline)
c_i	the time P_i is ready (completion time)



Half order of time constraints

For periodical processes:

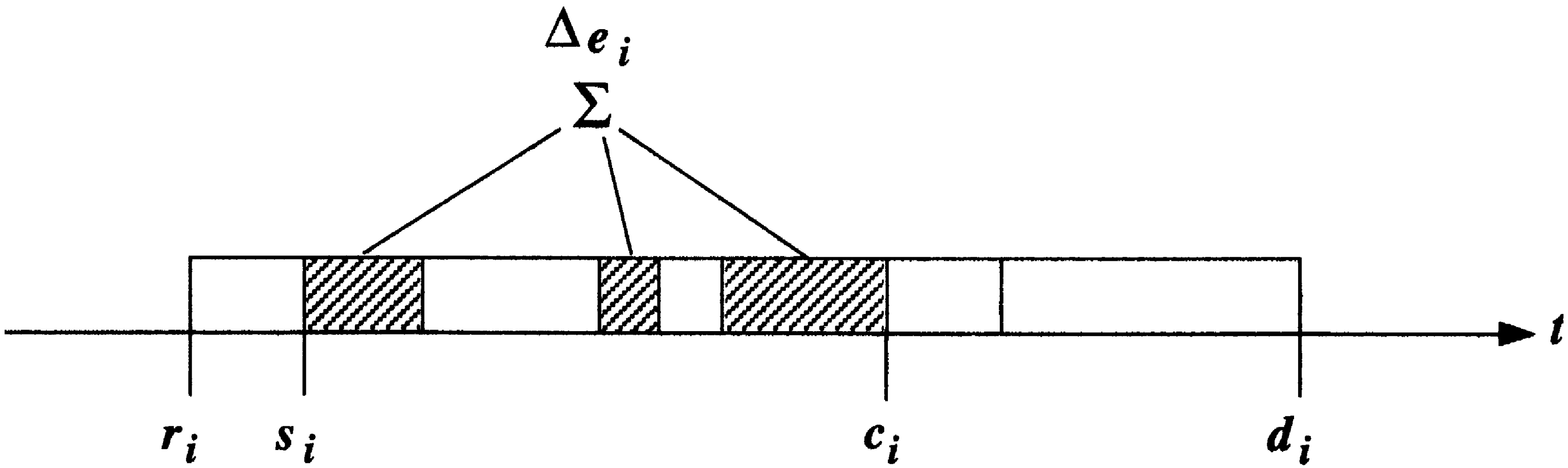
? p_i the time frame of a periodical process (period)

Process parameters for preemptive processes

For a non preemptive process P the following is true

$s_i + ? e_i = c_i$

which is **not** right for preemptive processes:



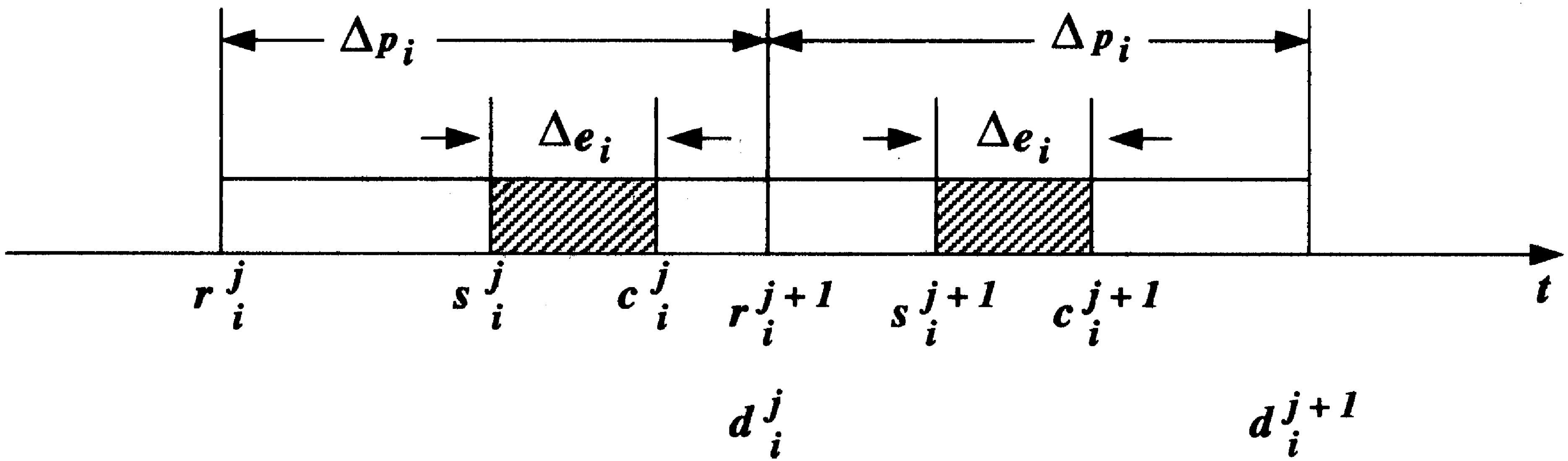
here we only get

$s_i + ? e_i ? c_i$

Process parameters for periodical processes

For periodical processes

? p_i is the time frame of a periodical process (period) defining ready times and deadlines for every repetition of that process:



RT scheduling is a complex task

Data (? e_i, r_i, s_i, d_i) may be known

- prior to the scheduling: **statical** RT scheduling
- during scheduling: **dynamical** RT scheduling

Results of RT scheduling

- complete plan: **explicite** RT scheduling
- plan rules: **implicite** RT scheduling

Phases within RT scheduling

- **feasibility check**
- schedule, **schedule construction**
- **dispatching**

Plans

A plan for a process set P={1,...,n} is a **practicable plan**, if by given ready times, execution times and deadlines the starting and completion times of all processes are selected so that

- no execution times overlap on a processor
- all time constraints are fulfilled

A statical (RT) scheduling method is called **optimal**, if it results for all process sets a practical plan if such one exists.
A dynamical (RT) scheduling method is called optimal, if it results for all process sets a practical plan if a statical (RT) scheduling method has found (afterwards, with knowledge of all ready times, execution times) one.

Planning methods

/Zöbel95/:

2 Basics of RT Planning

2.1 Process Model

2.2 Planning by Searching

2.3 EDF (Earliest Deadline First)

2.4 LLF (Least Laxity First)

2.5 Planning by Monotone Rates(*)

2.6 Evaluation of Planning Methods

(*)Priority for preemptive periodic processes

/Deinzer02/ (BSS1)

Non-preemptive Scheduling
- FCFS (First Come First Served)
- Suchen
- SJF (Shortest Job First)
- EDF (Earliest Deadline First)
- LLF (Least Laxity First)
- Priority

Preemptive Scheduling
- Round Robin
- SJF
- EDF
- Priority

Planning methods

- Non-preemptive Scheduling
  - FCFS (First Come First Served)
  - Searching
  - SJF (Shortest Job First)
  - EDF (Earliest Deadline First)
  - LLF (Least Laxity First)
  - Priority
- Preemptive Scheduling
  - Round Robin
  - SJF
  - EDF
  - Priority

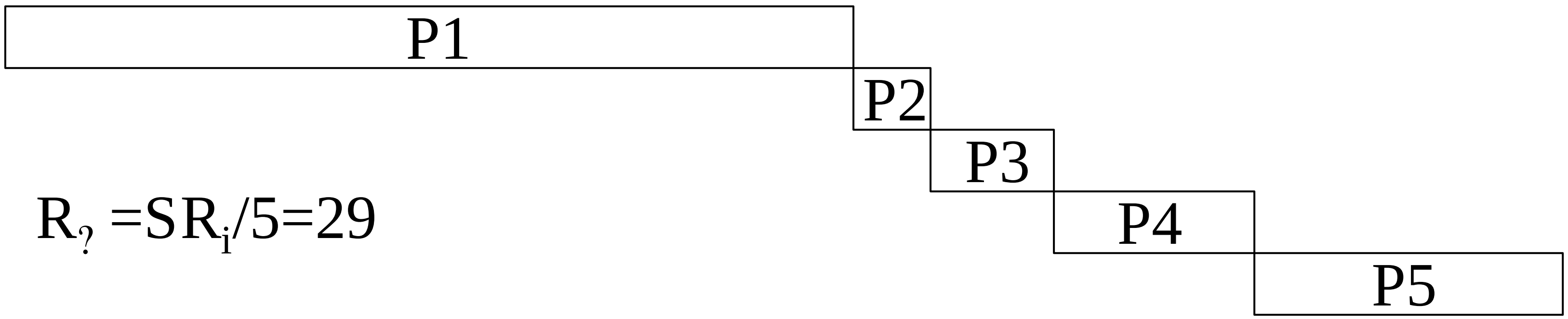
Planning methods

- Non-preemptive Scheduling
  - FCFS (First Come First Served)
  - Searching
  - SJF (Shortest Job First)
  - EDF (Earliest Deadline First)
  - LLF (Least Laxity First)
  - Priority
- Preemptive Scheduling
  - Round Robin
  - SJF
  - EDF
  - Priority

Non preemptive scheduling: FIFO (first in first out)

Processes are served in the sequence they enter the system.
(Queue, FIFO – First In, First Out).

<u>Example:</u>
Processes P_1, P_2, P_3, P_4 and P_5 enter the system in that sequence at t=0.
Execution times: $P_1=22$, $P_2=2$, $P_3=3$, $P_4=5$, $P_5=8$, all ready times $r_i=0$.
Results in answer(´Response´, completion, waiting) times ($R_i:=c_i-r_i$):
$R_1=22$, $R_2=22+2=24$, $R_3=24+3=27$, $R_4=27+5=32$, $R_5=32+8=40$.



$R_? = SR_i/5=29$

Planning methods

- Non-preemptive Scheduling
  - FCFS (First Come First Served)
  - Searching
  - SJF (Shortest Job First)
  - EDF (Earliest Deadline First)
  - LLF (Least Laxity First)
  - Priority
- Preemptive Scheduling
  - Round Robin
  - SJF
  - EDF
  - Priority

Non preemptive scheduling: by searching (w/o r_i, d_i)

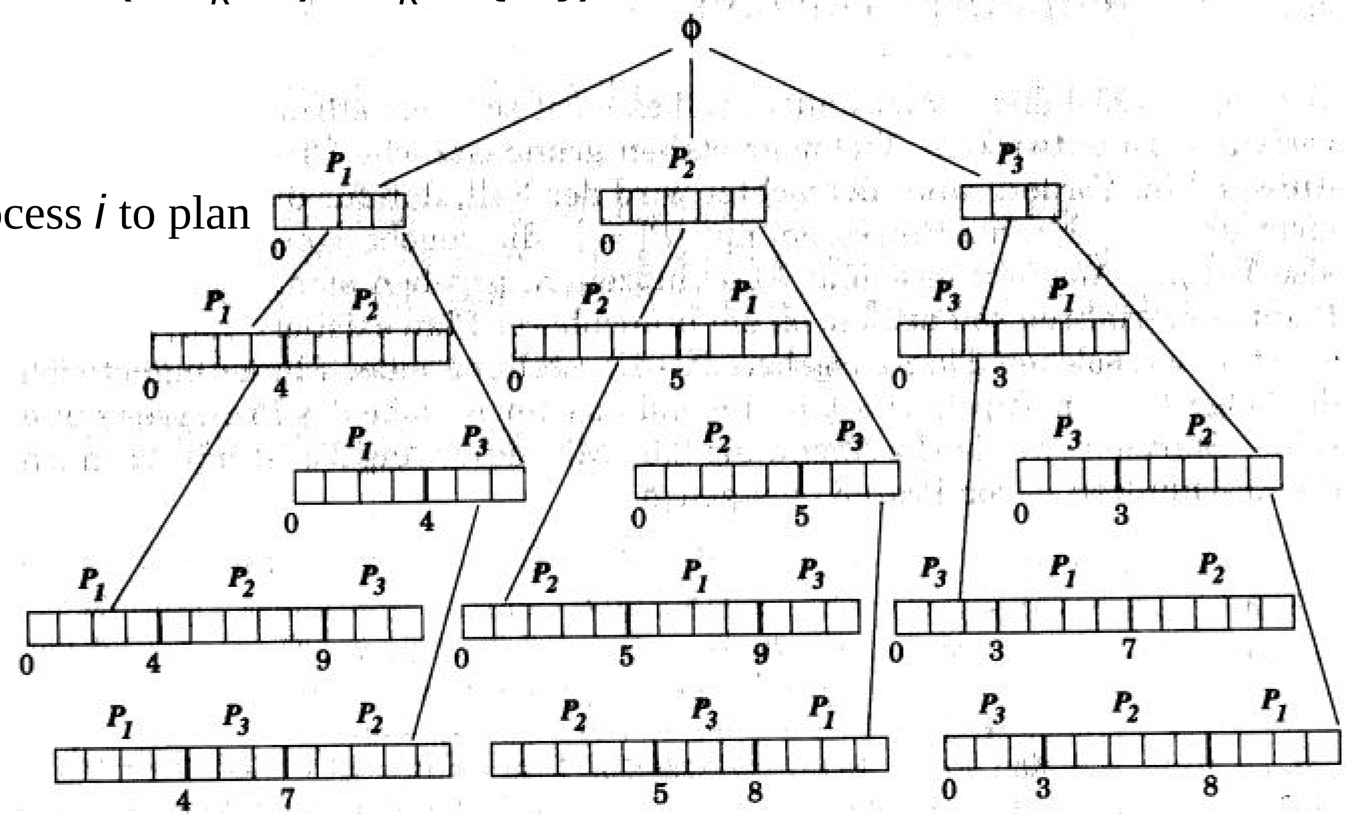
V1: *schedule*(PL_k, X_k):

FORALL i IN $P \setminus X_k$

schedule(*startPL* (PL_k, i), X_k ? { i })

Where
startPL(PL_k, i) ‘concatenates‘ process i to plan PL_k , result: plan PL_{k+1}

Without
- ready times
- deadlines



The leaves of the tree represent all sequences of 3 non preemptive processes

Non preemptive scheduling: by searching (with r_i, d_i)

V2: *schedule*(PL_k, X_k):

FORALL (i IN $P \setminus X_k$) AND *feasiblePL*(PL_k, i)

schedule(*earliestPL*(PL_k, i), X_k ? { i })

Where

feasiblePL(PL_k, i) tests whether process i can be integrated in PL_k (gap?)

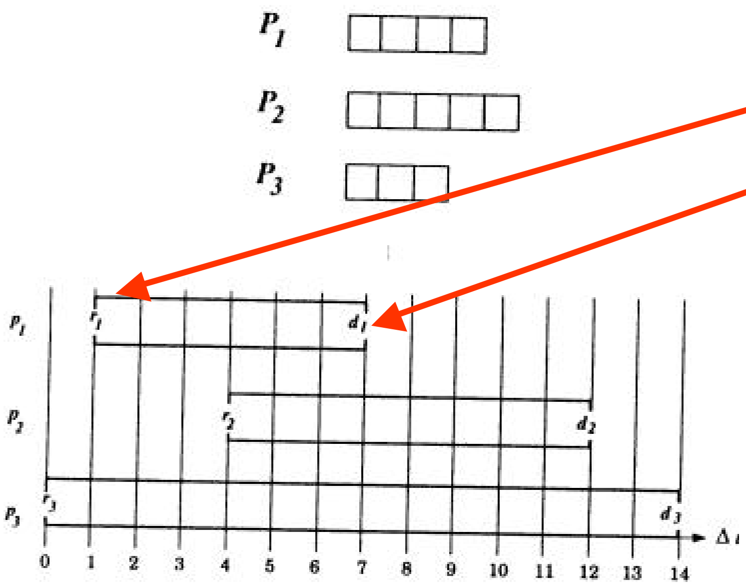
and

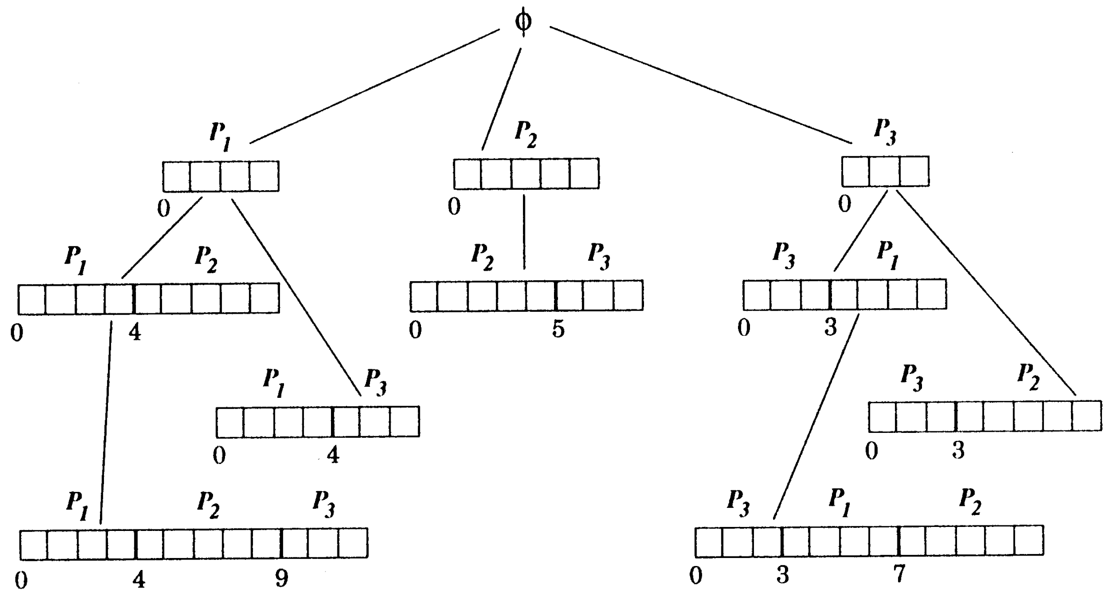
earliestPL(PL_k, i) uses plan PL_k and adds process i into earliest gap resulting in plan PL_{k+1}.

With

- ready times
- deadlines

Non preemptive scheduling: by searching (with r_i, d_i), example

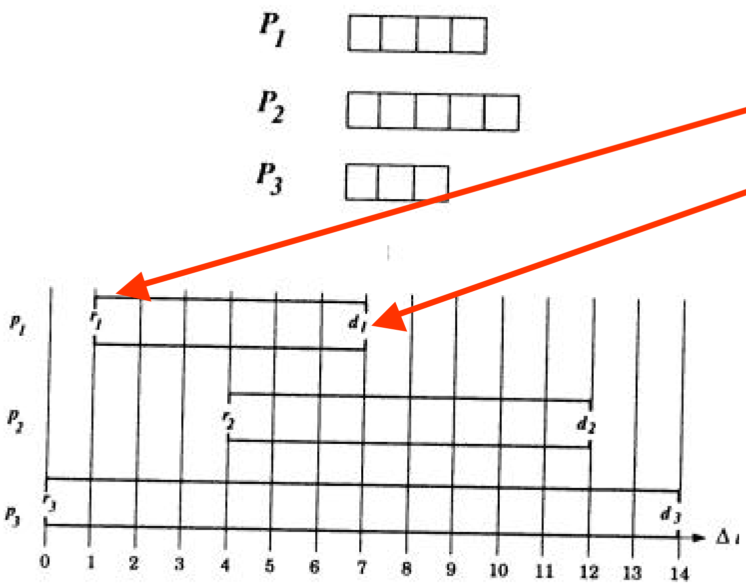


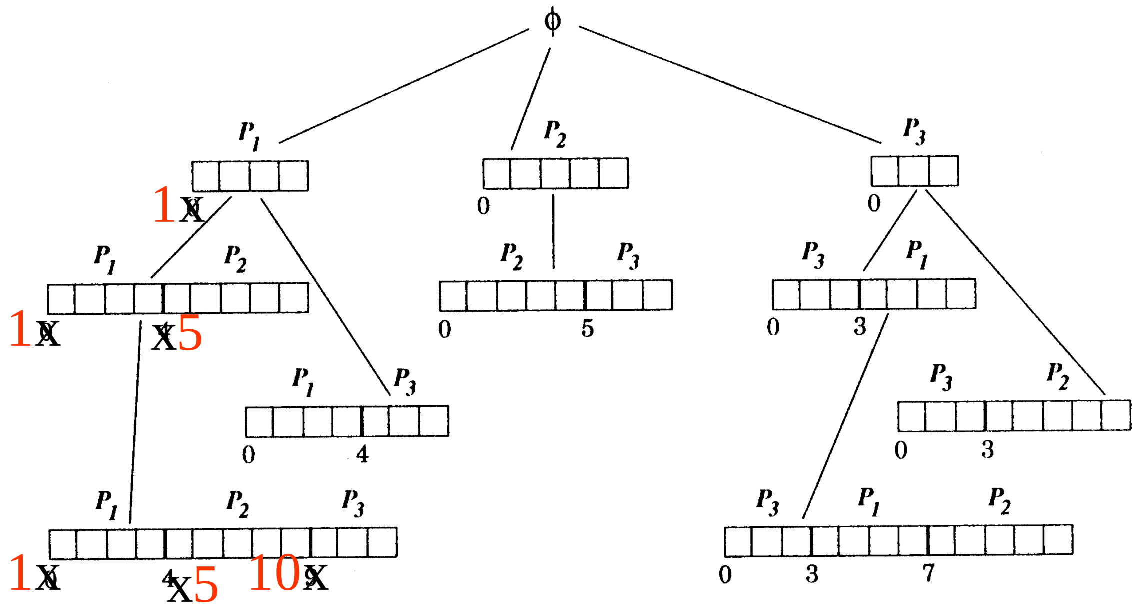


Which plans are practicable?

Are there other practicable plans?

Non preemptive scheduling: by searching (with r_i, d_i), example





Which plans are practicable?

Are there other practicable plans?

Planning methods

- Non-preemptive Scheduling
  - FCFS (First Come First Served)
  - Searching
  - SJF (Shortest Job First)
  - EDF (Earliest Deadline First)
  - LLF (Least Laxity First)
  - Priority
- Preemptive Scheduling
  - Round Robin
  - SJF
  - EDF
  - Priority

Non preemptive scheduling: SJF (Shortest Job First)

Process with shortest execution time gets CPU first (if equal: FCFS).

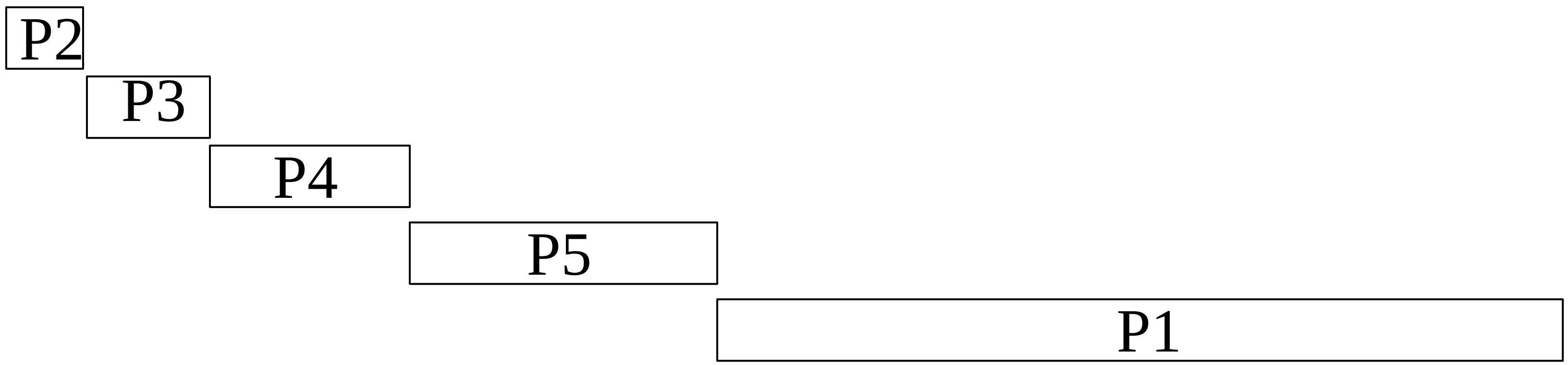
<u>Example:</u>

Execution times: P_1=22, P_2=2, P_3=3, P_4=5, P_5=8, all ready times r_i=0.

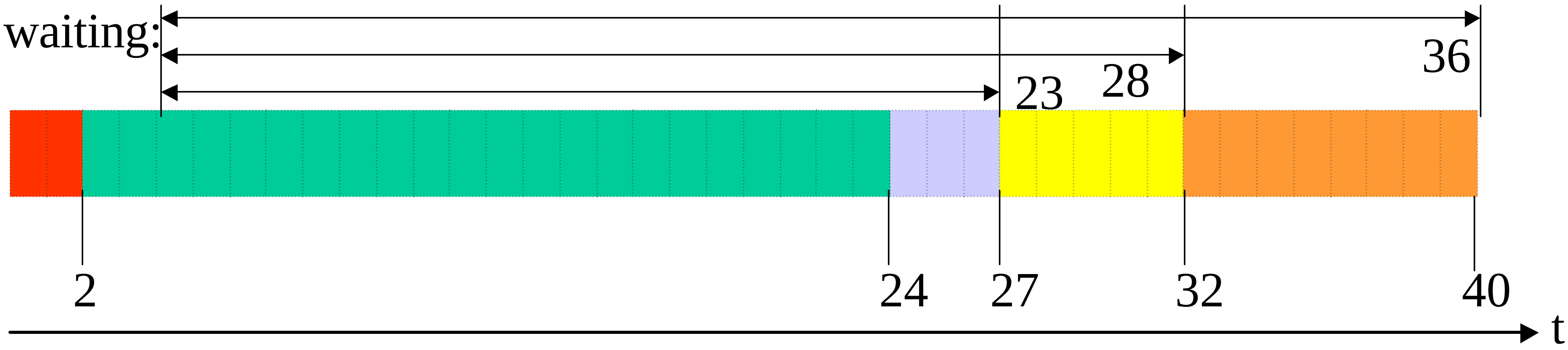
Response times:

R_1=40, R_2=2, R_3=5, R_4=10, R_5=18.

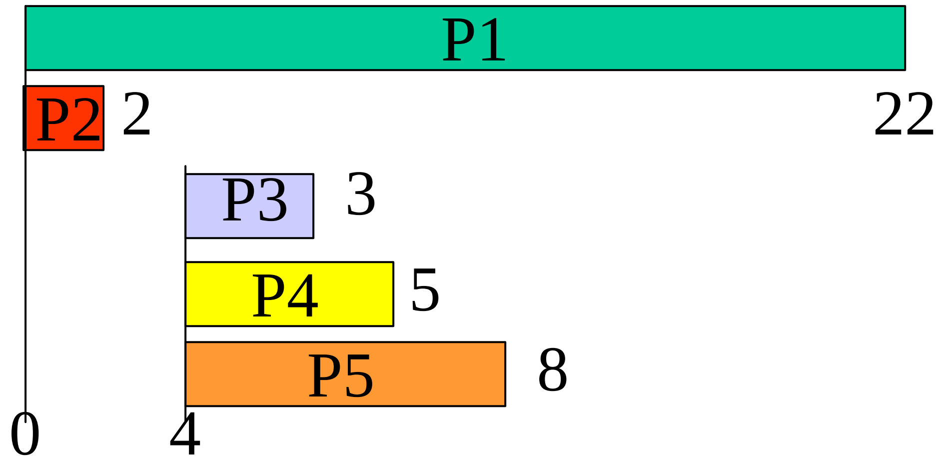
$R_? = SR_i/5 = 15$



Non preemptive scheduling: SJF (example)



$R_{?} = SR_i/5=(24+2+23+28+36)/5=113/5=22,6$



<u>Example</u>:

Execution times:

P_1=22, P_2=2, P_3=3, P_4=5, P_5=8,

Ready times:

P_1=0, P_2=0, P_3=4, P_4=4, P_5=4.

Planning methods

- Non-preemptive Scheduling
  - FCFS (First Come First Served)
  - Searching
  - SJF (Shortest Job First)
  - EDF (Earliest Deadline First)
  - LLF (Least Laxity First)
  - Priority
- Preemptive Scheduling
  - Round Robin
  - SJF
  - EDF
  - Priority

Non preemptive scheduling: EDF (Earliest Deadline First)

Processes P = {1,...,n} are already ordered by their deadlines:
1<=i<=j<=n ? $d_i <= d_j$

V3: *schedule*(*PL,P*):
PL = <>;
i = 1;
WHILE (*i* <= n) AND feasible*PL*(*PL, i*)
BEGIN
PL = *deadlinePL* (*PL*??*i*);
i = *i* +1;
END

Where:
deadlinePL(PL_k, *i*) makes from given plan PL_k by introducing of process *i* (with smallest deadline of all runnable processes) plan PL_{k+1}.

Planning methods

- Non-preemptive Scheduling
  - FCFS (First Come First Served)
  - Searching
  - SJF (Shortest Job First)
  - EDF (Earliest Deadline First)
  - LLF (Least Laxity First)
  - Priority
- Preemptive Scheduling
  - Round Robin
  - SJF
  - EDF
  - Priority

Non preemptive scheduling: LLF (Least Laxity First)

Select from the runnable processes that process, who´s laxity

$? \, lax_i = (d_i - r_i) - ? \, e_i$

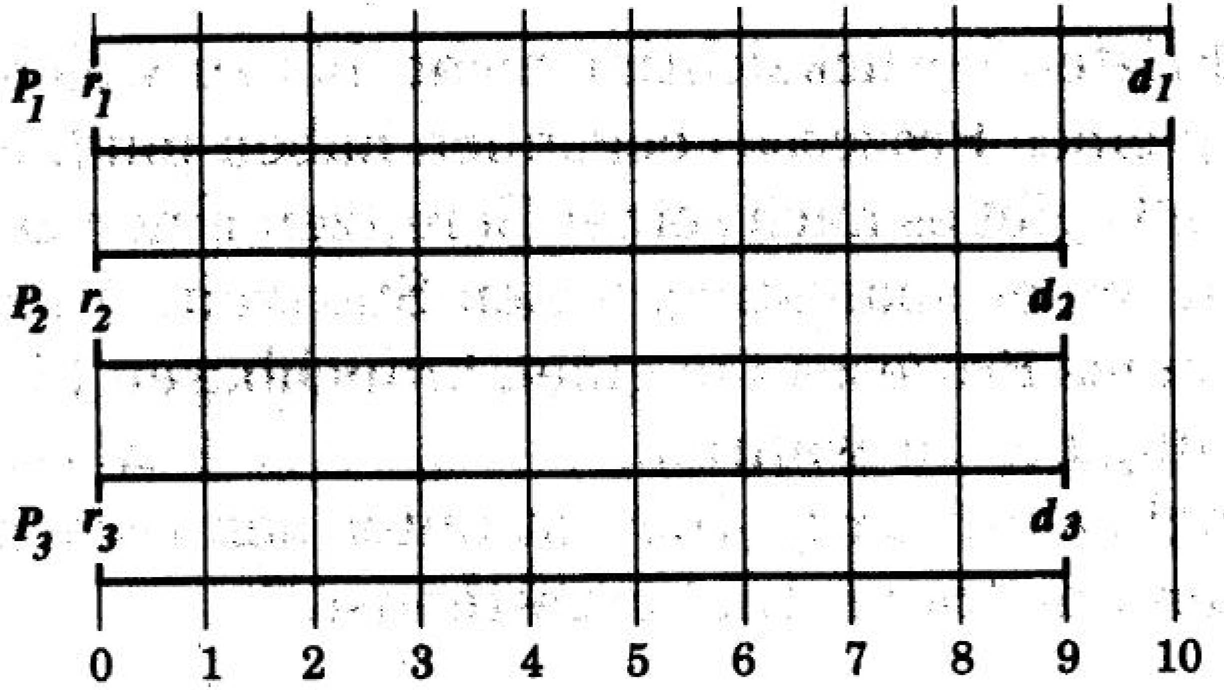
is the shortest.

Non preemptive scheduling: LLF vs. EDF

Example:

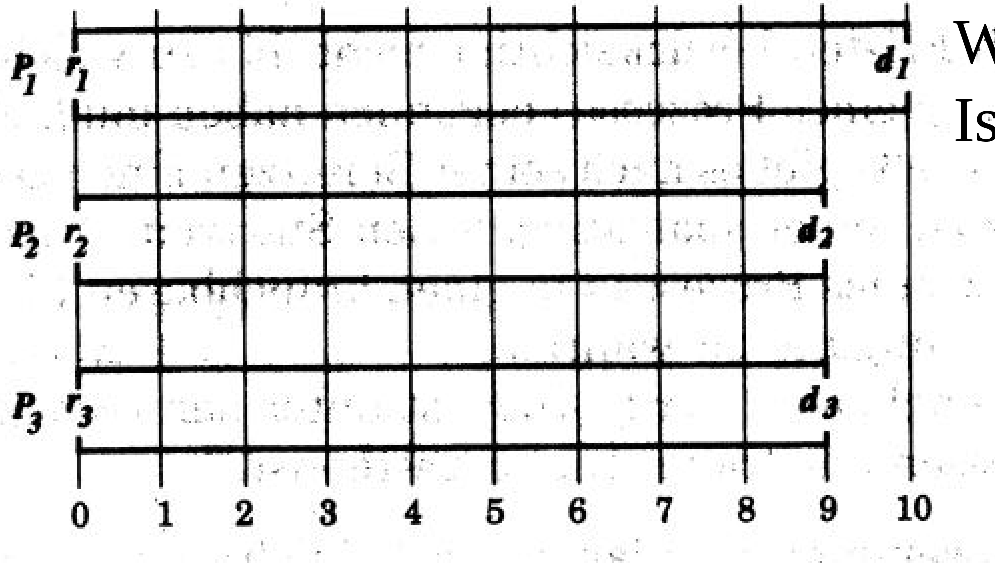
Given 2 CPUs and non preemptive processes P = {1, 2, 3} with:

	r_i	d_i	? e_i
i=1	0	10	8
i=2	0	9	5
i=3	0	9	4

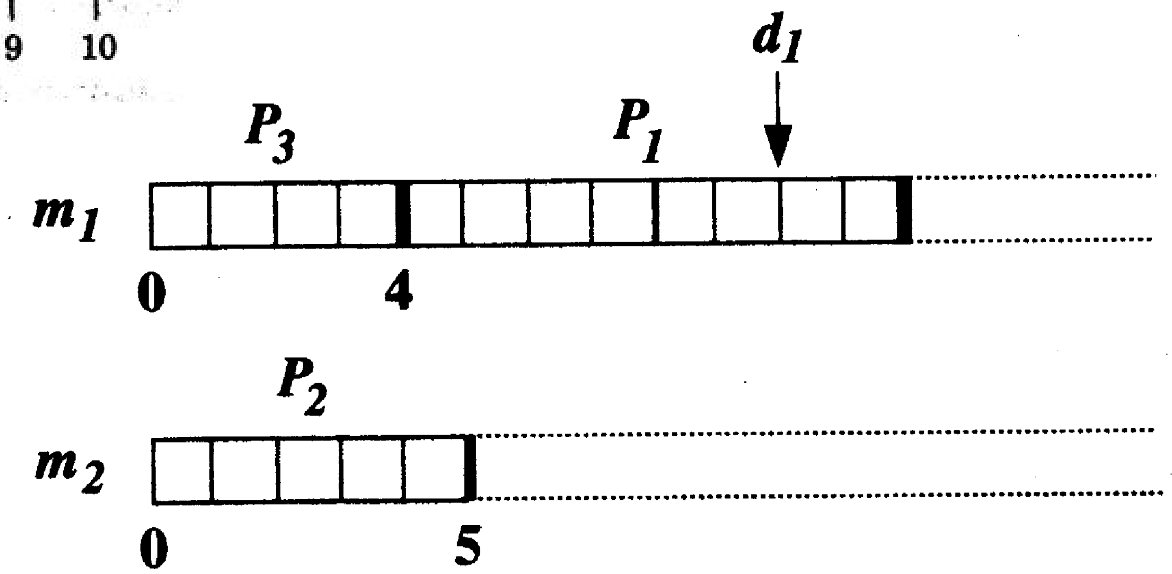


What happens by planning according EDF? Is there a practicable plan?
What happens by planning according LLF? Is there a practicable plan?

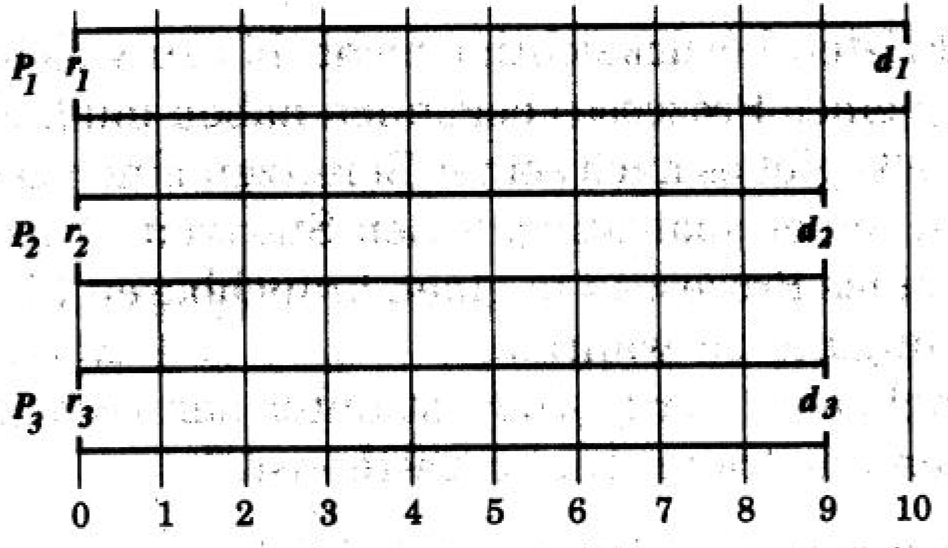
Non preemptive scheduling: LLF vs. EDF



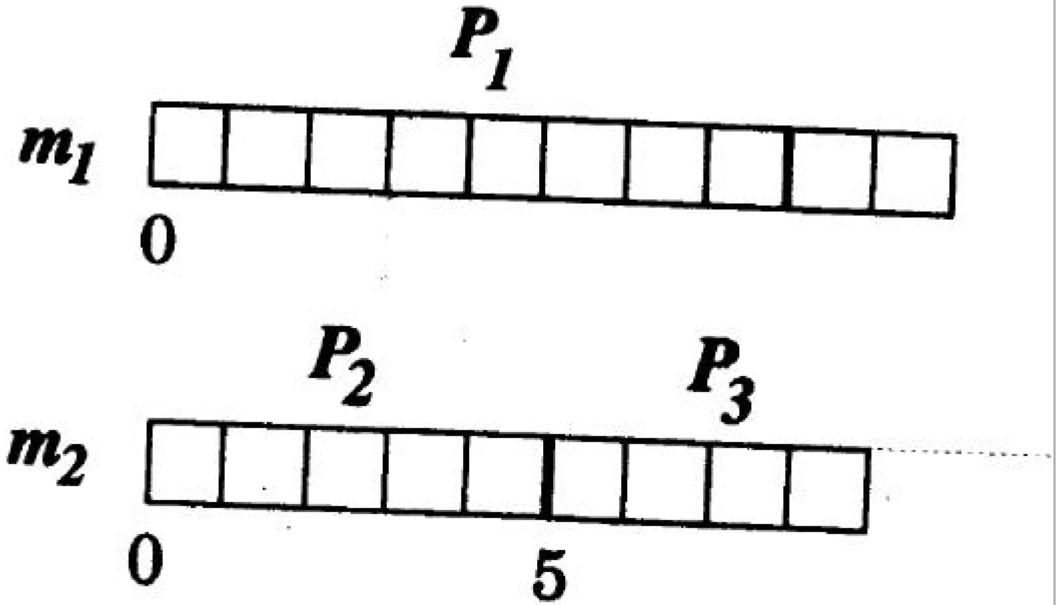
What happens by planning according EDF?
Is there a practicable plan?



Non preemptive scheduling: LLF vs. EDF



What happens by planning according LLF?
Is there a practicable plan?

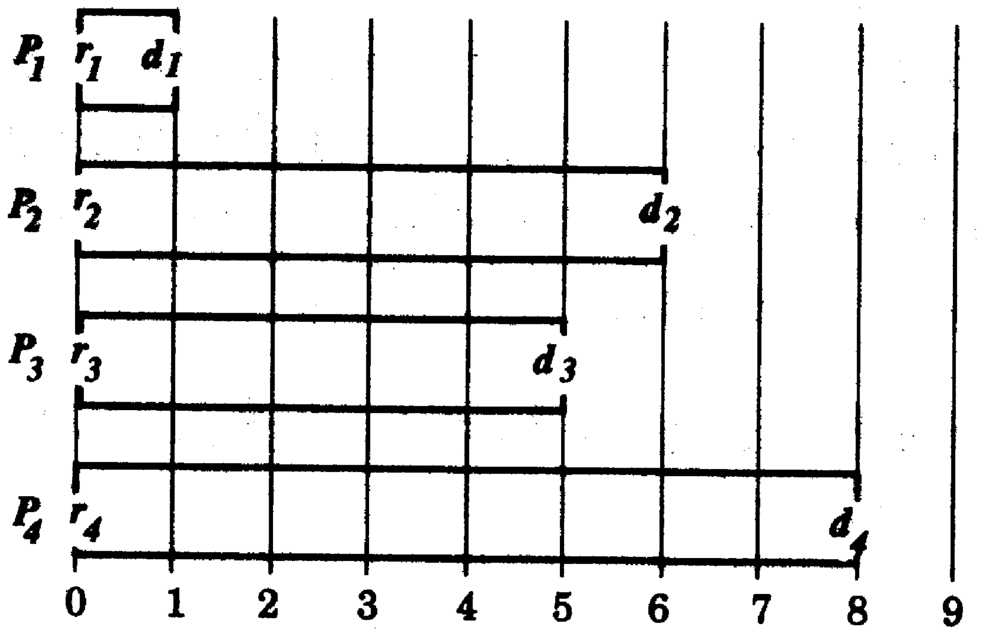


Non preemptive scheduling: LLF - is it optimal?

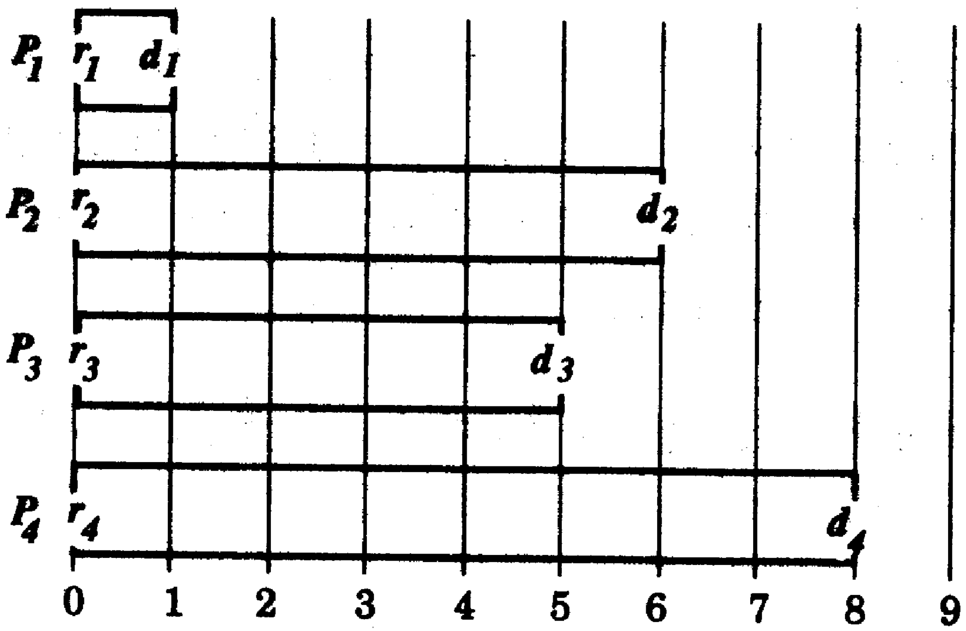
Example:

Given 2 CPUs and non preemptive processes P = {1, 2, 3, 4} with:

	r_i	d_i	? e_i
i=1	0	1	1
i=2	0	6	5
i=3	0	5	3
i=4	0	8	5



Non preemptive scheduling: LLF - is it optimal?



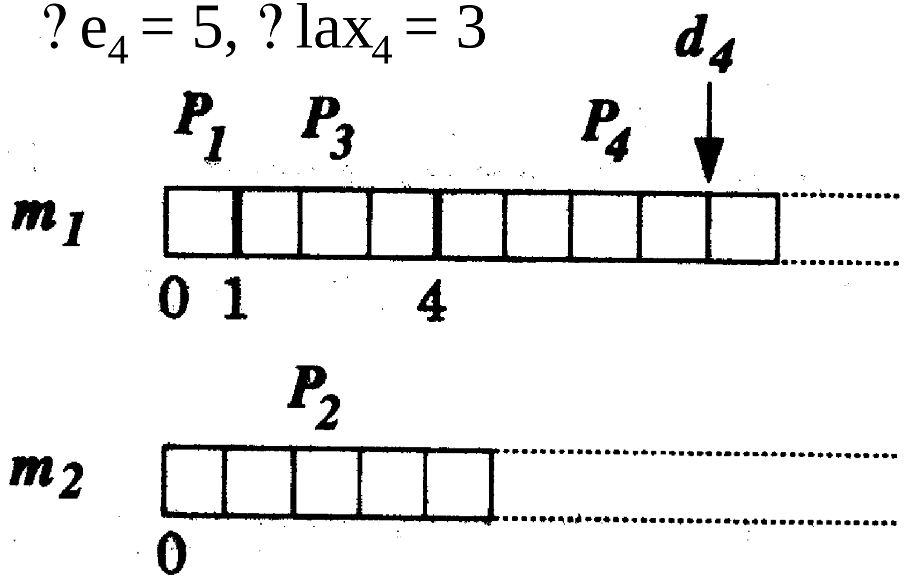
? $e_1 = 1$, ? $lax_1 = 0$

? $e_2 = 5$, ? $lax_2 = 1$

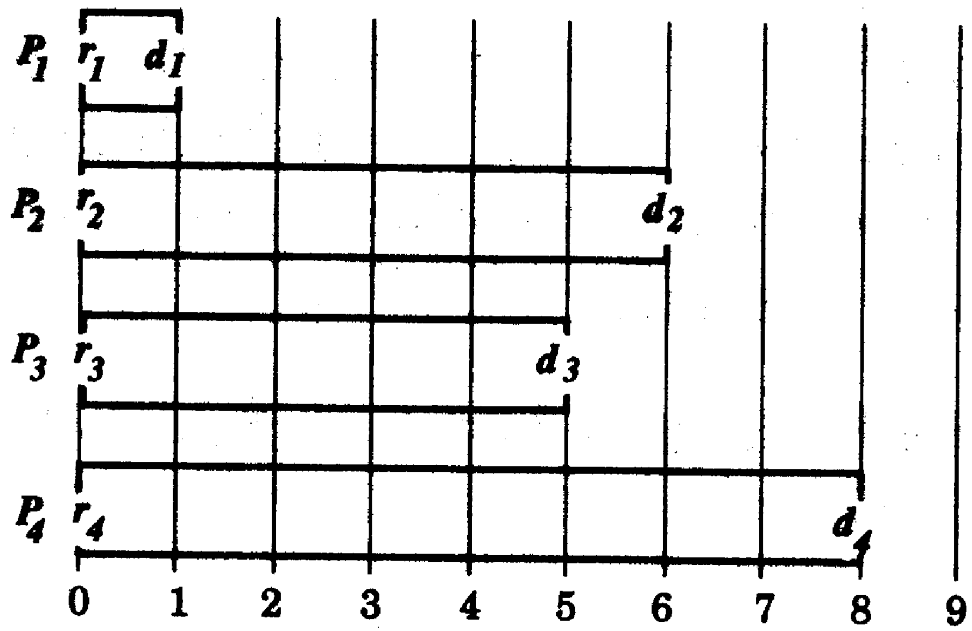
? $e_3 = 3$, ? $lax_3 = 2$

? $e_4 = 5$, ? $lax_4 = 3$

Plan by LLF is **not** practicable:



Non preemptive scheduling: LLF - is it optimal?



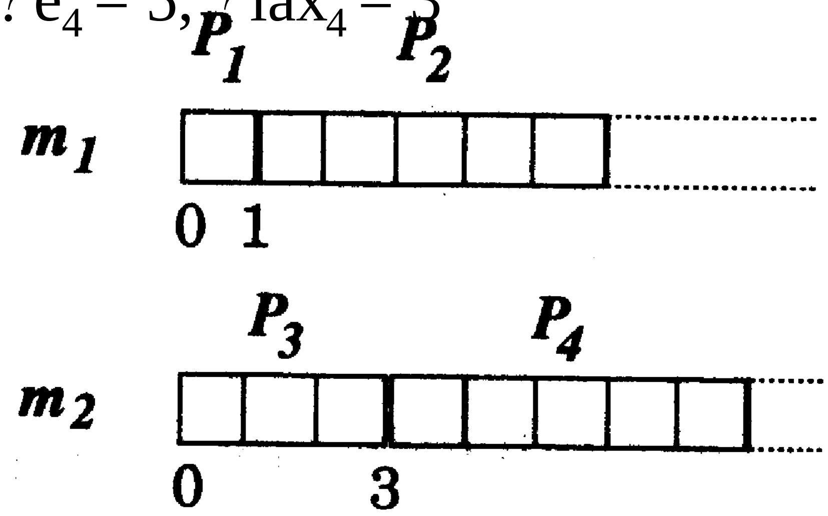
? $e_1 = 1$, ? $lax_1 = 0$

? $e_2 = 5$, ? $lax_2 = 1$

? $e_3 = 3$, ? $lax_3 = 2$

? $e_4 = 5$, ? $lax_4 = 3$

But there **is** a practicable plan:



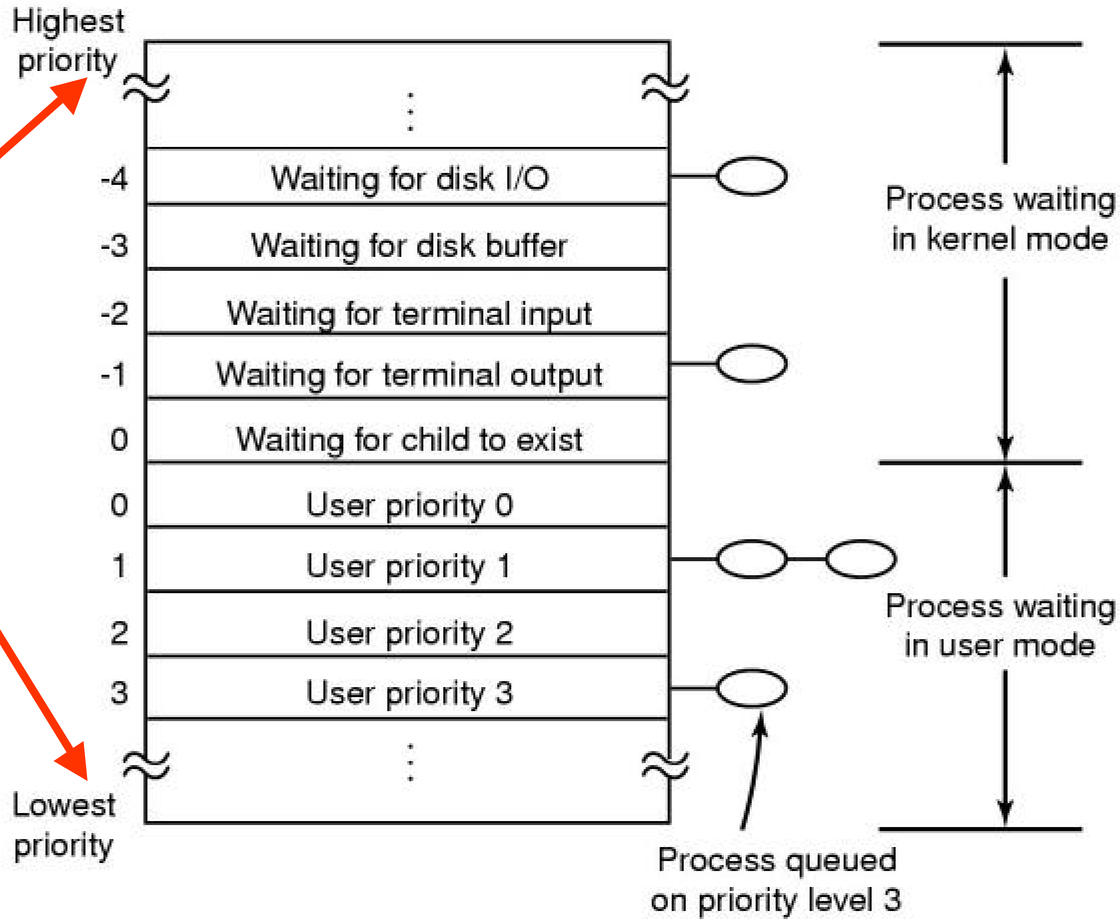
So: **LLF is not optimal!**

Planning methods

- Non-preemptive Scheduling
  - FCFS (First Come First Served)
  - Searching
  - SJF (Shortest Job First)
  - EDF (Earliest Deadline First)
  - LLF (Least Laxity First)
  - Priority
- Preemptive Scheduling
  - Round Robin
  - SJF
  - EDF
  - Priority

Non preemptive scheduling: priority

Processes are classified into different ´urgent´ classes. First the processes with the ´most urgent´ class (highest **priority** – most times (Unix, NT,...) lowest number) get the CPU, than ´next urgent´ ... Priorities may be defined **internal**, (i.e by the OS) **external**, **statical** or **dynamical** (? Feedback Scheduling).



Example: Unix Scheduler (priority based waiting queues)
/Tanenbaum02/

Planning methods

- Non-preemptive Scheduling
  - FCFS (First Come First Served)
  - Searching
  - SJF (Shortest Job First)
  - EDF (Earliest Deadline First)
  - LLF (Least Laxity First)
  - Priority
- Preemptive Scheduling
  - Round Robin
  - SJF
  - EDF
  - Priority

Non preemptive vs. preemptive scheduling

Non-preemptive scheduling strategies are not appropriate for dialog systems and may definitely not used for RT systems.

You easily find (counter) examples!

So the OS has the right to take away the CPU from a process in preemptive scheduling strategies.
This mechanism is also used on a regular base.

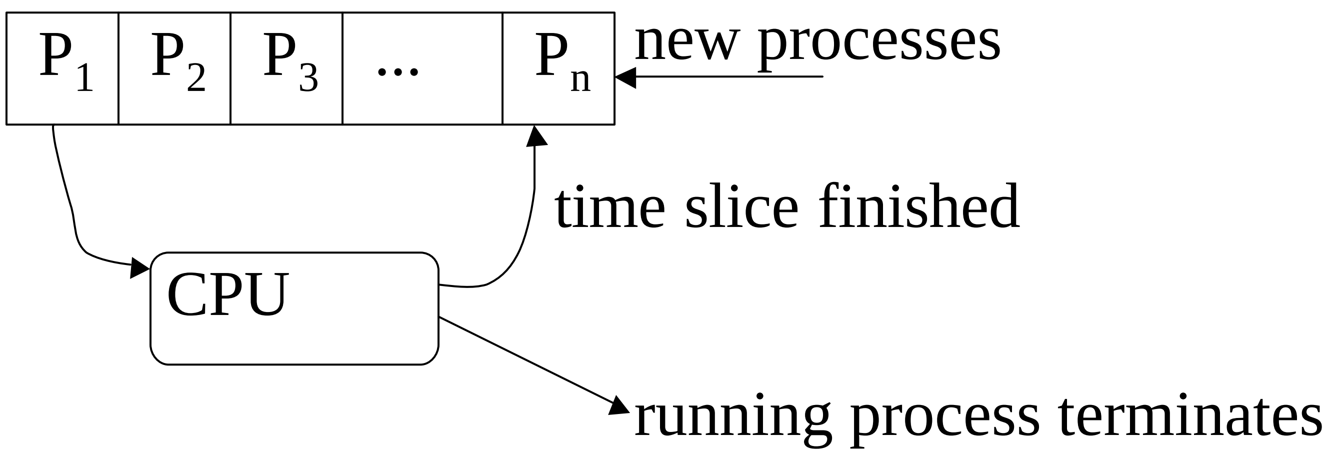
Most scheduling strategies for non preemptive systems can be used in preemptive systems too.

Planning methods

- Non-preemptive Scheduling
  - FCFS (First Come First Served)
  - Searching
  - SJF (Shortest Job First)
  - EDF (Earliest Deadline First)
  - LLF (Least Laxity First)
  - Priority
- Preemptive Scheduling
  - Round Robin
  - SJF
  - EDF
  - Priority

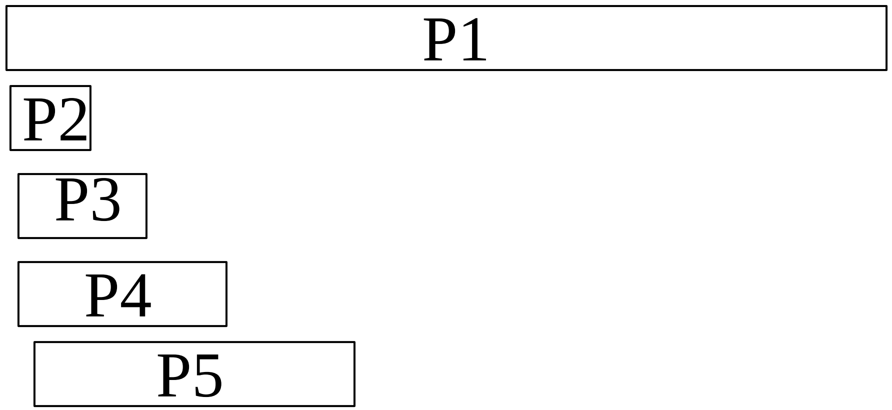
Preemptive scheduling: round robin

Ready processes wait – as FCFS – in a waiting queue.
First process gets CPU – but only for a certain amount of time (**time slice**, quantum); if process doesn´t finish in that time slice he will be interrupted and put at the end of the waiting queue.



Preemptive scheduling: round robin (example)

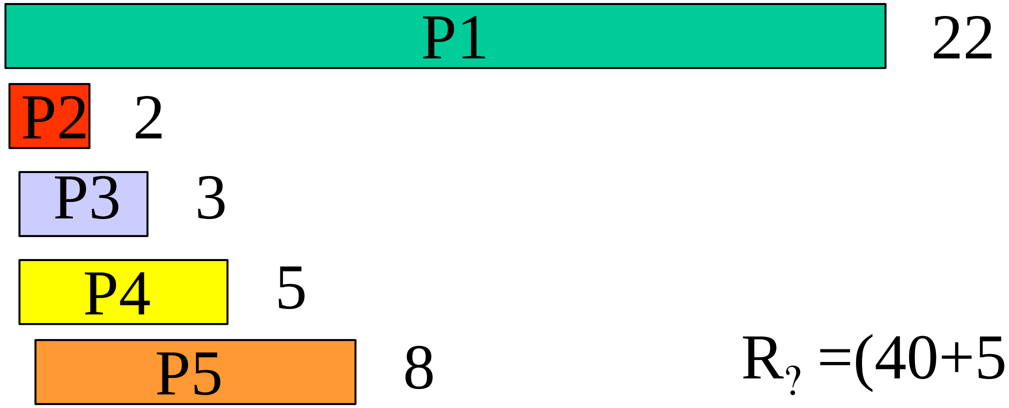
Execution times: P_1=22, P_2=2, P_3=3, P_4=5, P_5=8.



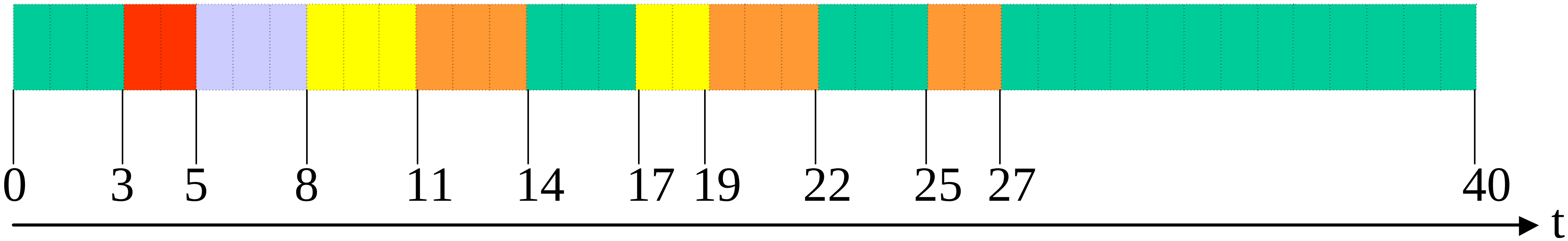
Quantum 3 time units, switching time (not realistic!) 0 time units.

- What´s the sequence of the processes?
- What´s the average response time?

Preemptive scheduling: round robin (example)



$R_{?} = (40+5+8+19+27)/5 = 99/5 = 19{,}8$



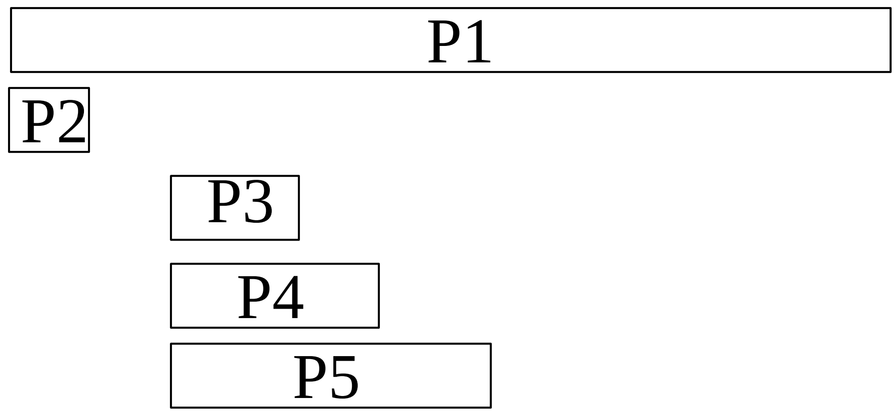
Planning methods

- Non-preemptive Scheduling
  - FCFS (First Come First Served)
  - Searching
  - SJF (Shortest Job First)
  - EDF (Earliest Deadline First)
  - LLF (Least Laxity First)
  - Priority
- Preemptive Scheduling
  - Round Robin
  - SJF
  - EDF
  - Priority

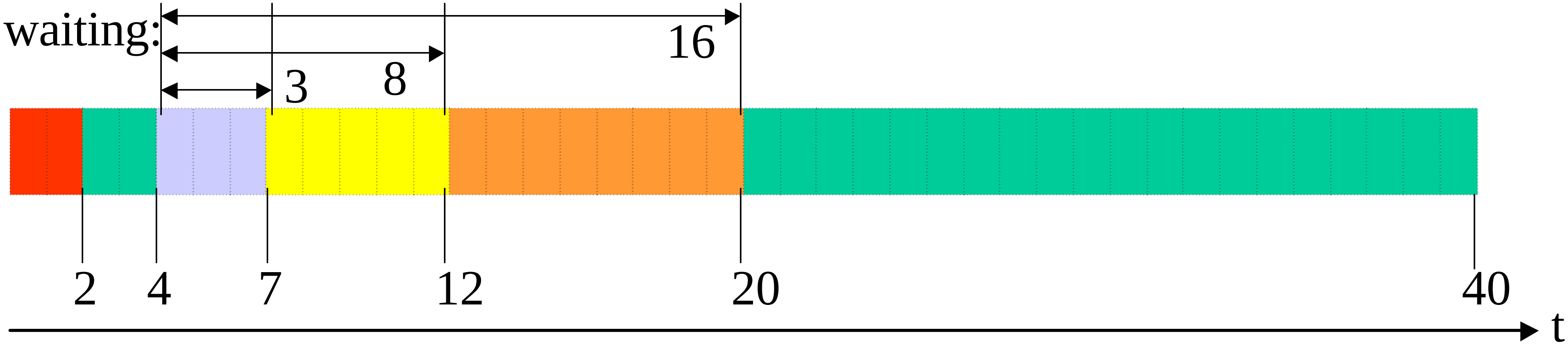
Preemptive scheduling: SJF

Most non preemptive strategies can be used preemptive too, e.g. SJF, priority scheduling or EDF (earliest deadline first).
At the preemptive version of SJF the running process is stopped as soon as a new (ready) process enters the system. Now the scheduler calculates, which process has the smalles (remaining!) execution time and selects this one for the processor.

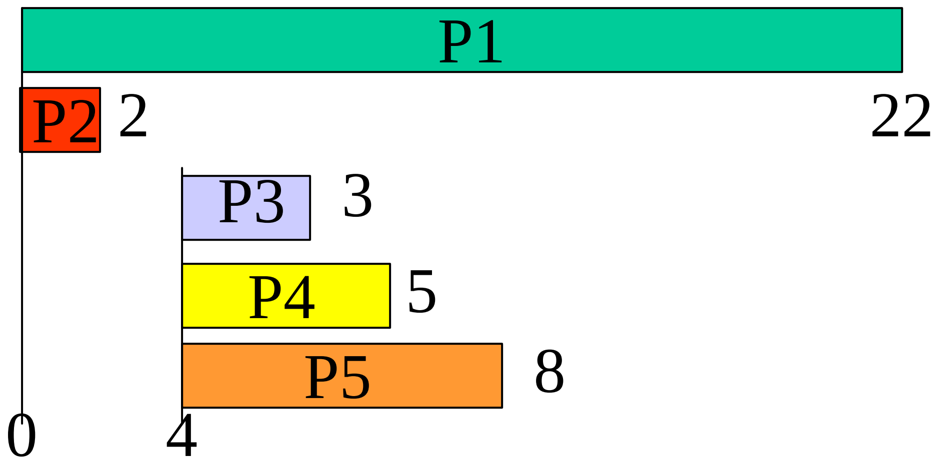
Example:
Execution times:
P_1=22, P_2=2, P_3=3, P_4=5, P_5=8,
Ready times:
P_1=0, P_2=0, P_3=4, P_4=4, P_5=4.



Preemptive scheduling: SJF (example)

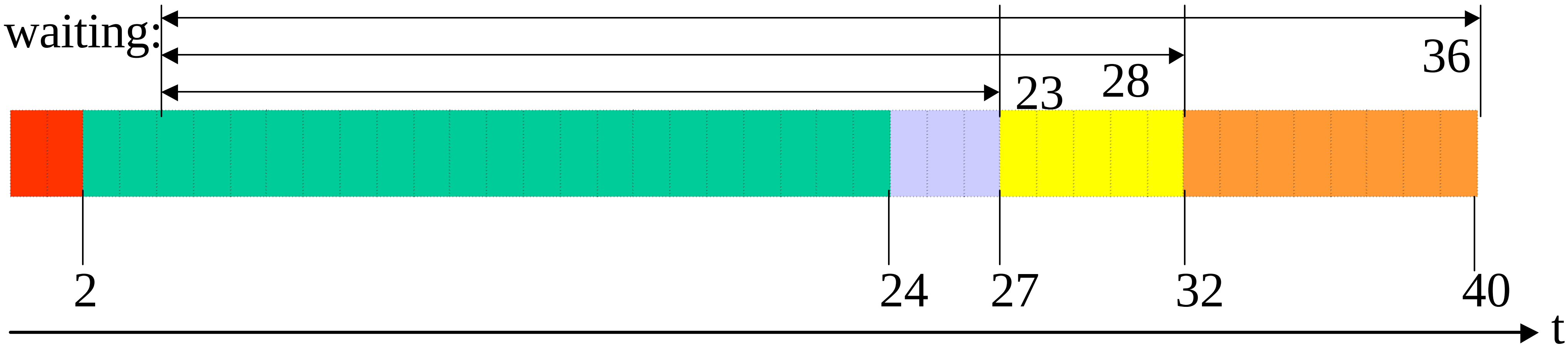


$R_? = SR_i/5=(40+2+3+8+16)/5=69/5=13,8$



Compare completion times (and their average) with non preemptive SJF!

Preemptive vs non preemptive scheduling: SJF (example)



$R_{?} = \Sigma R_i/5 = (24+2+23+28+36)/5 = 113/5 = 22{,}6$

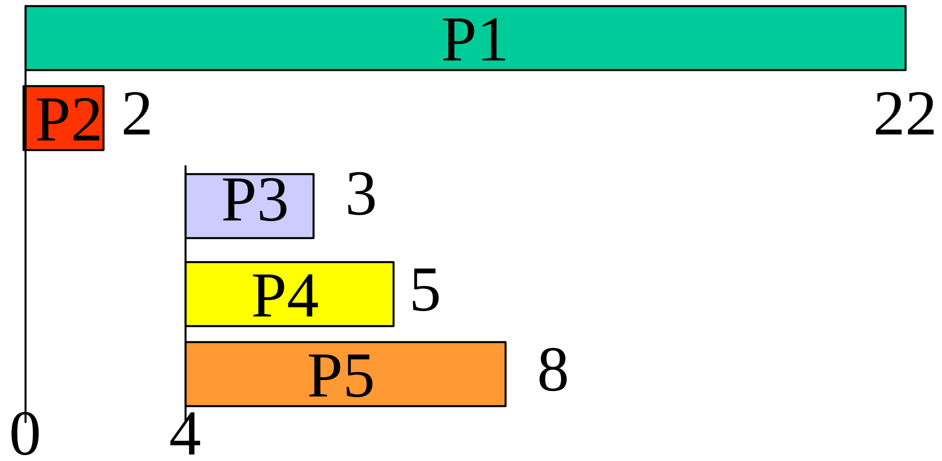
<u>Example</u>:

Execution times:

$P_1=22$, $P_2=2$, $P_3=3$, $P_4=5$, $P_5=8$,

Ready times:

$P_1=0$, $P_2=0$, $P_3=4$, $P_4=4$, $P_5=4$.



Planning methods

- Non-preemptive Scheduling
  - FCFS (First Come First Served)
  - Searching
  - SJF (Shortest Job First)
  - EDF (Earliest Deadline First)
  - LLF (Least Laxity First)
  - Priority
- Preemptive Scheduling
  - Round Robin
  - SJF
  - EDF
  - Priority

Preemptive scheduling: EDF

Remember how we´ve done it **non** preemptive:

Non preemptive scheduling: EDF (Earliest Deadline First)

Processes P = {1,...,n} are already ordered by their deadlines:
$1<=i<=j<=n$? $d_i <= d_j$

```
V3: schedule( PL,P ):
PL = <>;
i = 1;
WHILE ( i <= n ) AND feasiblePL(PL, i )
BEGIN
PL = deadlinePL (PL??i );
i = i +1;
END
```

Where:

deadlinePL(PL_k, *i*)	makes from given plan PL_k by introducing of process *i* (with smallest deadline of all runnable processes) plan PL_{k+1}.

Realtime Systems – **RT Scheduling**
Arnulf Deinzer, FH Kempten, Winter Term 2003/2004
2.23

Preemptive scheduling: EDF

Processes P = {1,...,4} can be interrupted (i.e. we have a preemptive OS):

	r_i	d_i	? e_i
i=1	0	5	4
i=2	0	7	1
i=3	0	7	2
i=4	0	13	5

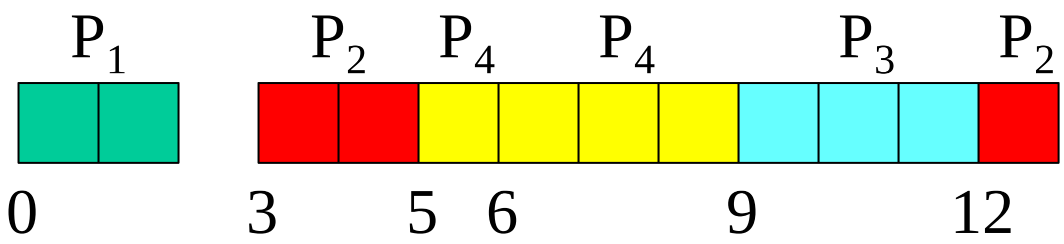
Try to adapt V3 (EDF, non preemptive) for the preemptive case!
Work your algorithm on processes above!

Preemptive scheduling: EDF

```
V4: Schedule(PL,P):
    PL=<>;
    t=min {r i|i? P}
    WHILE ? allinPL(t) DO
        IF Ready(t)=<> THEN t= nextavail(t);
        ELSE
        BEGIN
            i=edf(Ready(f));
            IF ?  feasible(i,t) THEN BREAK;
            ? l=min(rest(i,t), nextavail(t)-t);
            PL=PL^(i,t, ? l);
            t=t+? l;
        END;
```

Example:

	r_i	d_i	? e_i
i=1	0	4	2
i=2	3	14	3
i=3	6	12	3
i=4	5	10	4



Planning methods

- Non-preemptive Scheduling
  - FCFS (First Come First Served)
  - Searching
  - SJF (Shortest Job First)
  - EDF (Earliest Deadline First)
  - LLF (Least Laxity First)
  - Priority
- Preemptive Scheduling
  - Round Robin
  - SJF
  - EDF
  - Priority

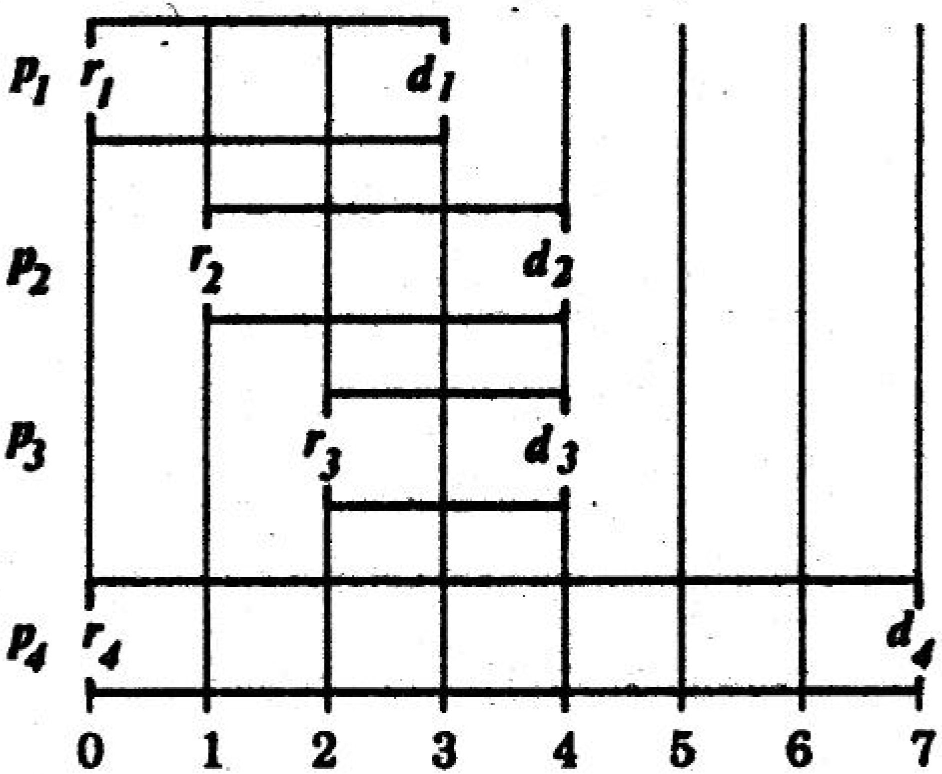
Preemptive scheduling: priority

Most non preemptive strategies can be used preemptive too, e.g. SJF, priority scheduling or EDF (earliest deadline first).
At the preemptive version of priority scheduling the running process is stopped as soon as a new (ready) process enters the system. Now the scheduler checks, which process has the highest priority (which may the smallest number!) and selects this one for the processor.

Exercise:
Given a preemptive system with processes P_1, P_2, P_3, P_4, P_5 which enter in that sequence at the same time (sorry!) the system. They have execution times $P_1=15$, $P_2=7$, $P_3=1$, $P_4=4$, $P_5=8$.
What is the average response time if as strategy

- FCFS
- SJF
- Round Robin with Q=4 is selected?

Preemptive scheduling: planing game(1)



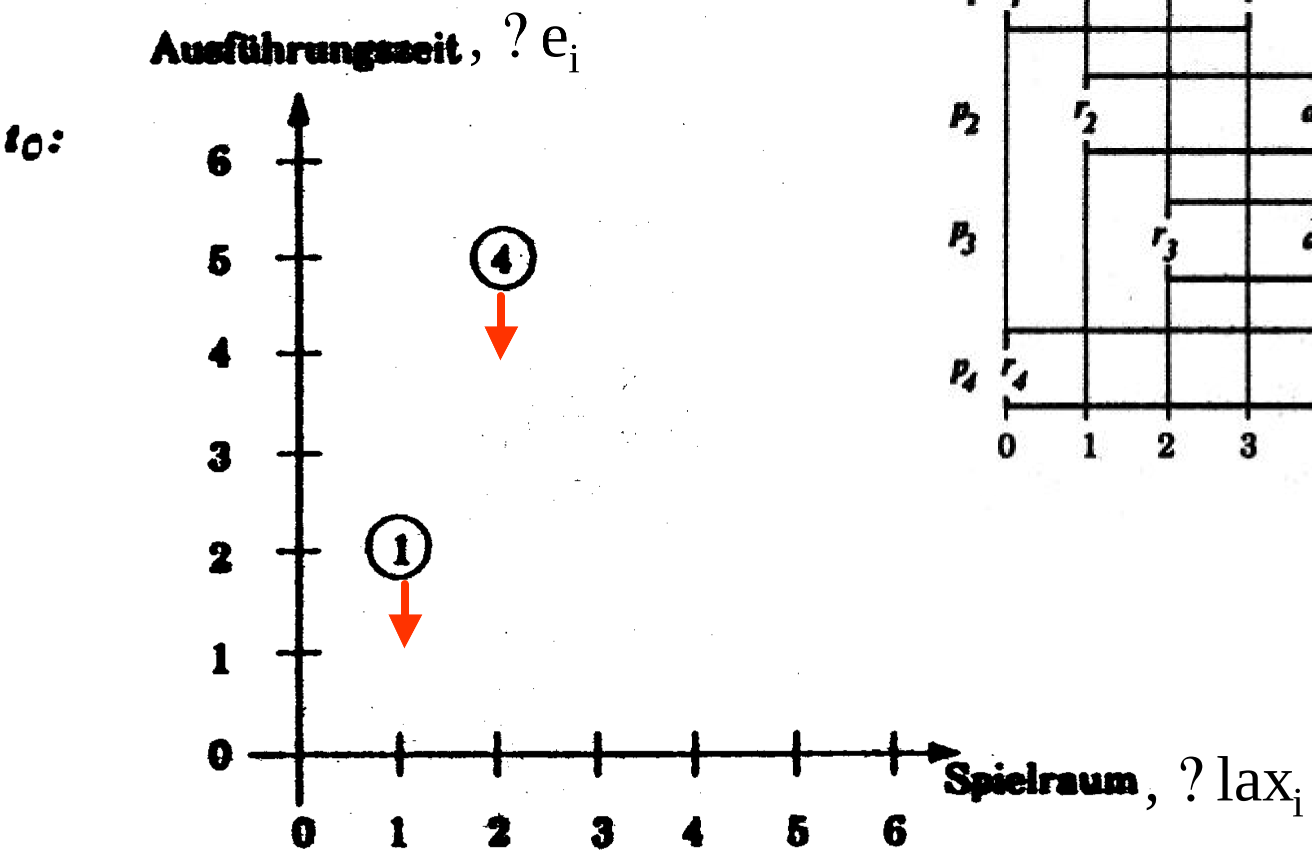
Example:

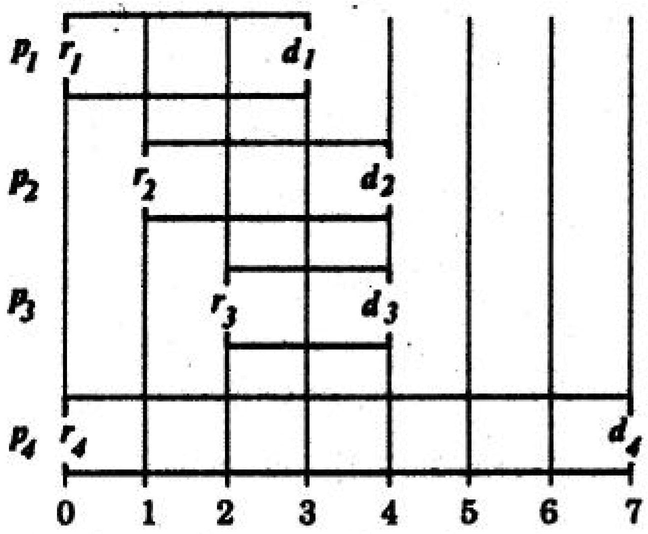
	r_i	d_i	? e_i
i=1	0	3	2
i=2	1	4	3
i=3	2	4	1
i=4	0	7	5

2 CPUs

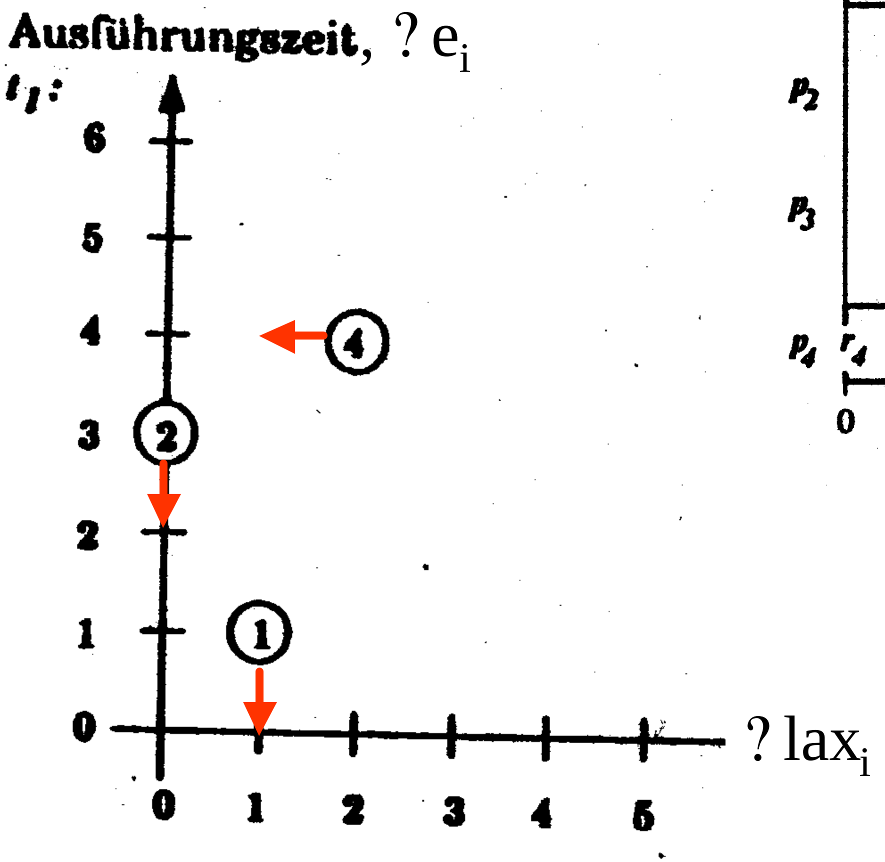
LLF with preemptive OS/processes

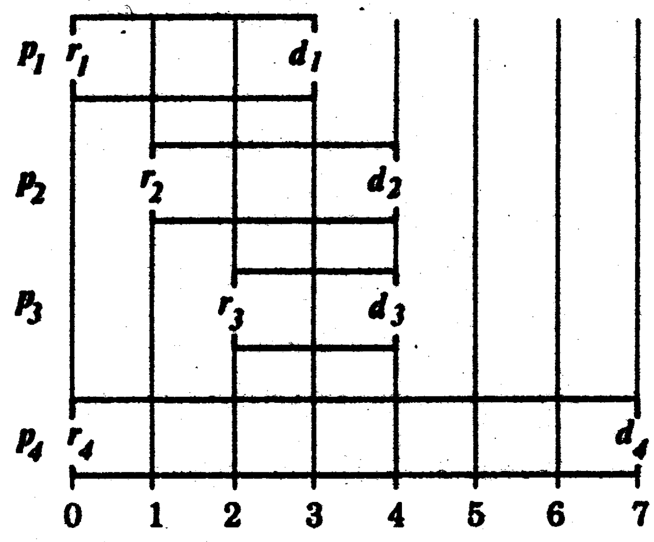
Preemptive scheduling: planing game(2, t=0)



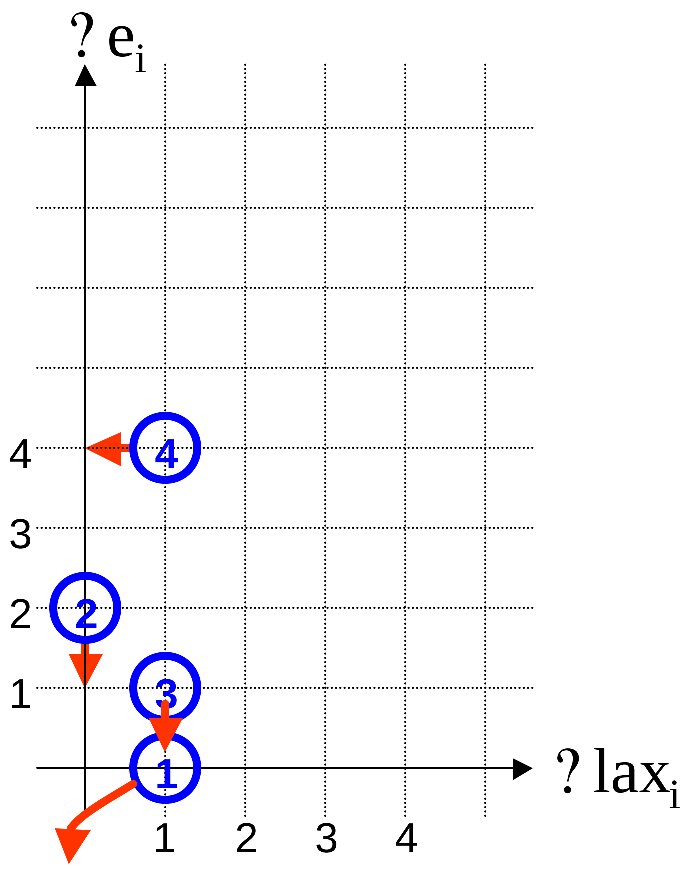


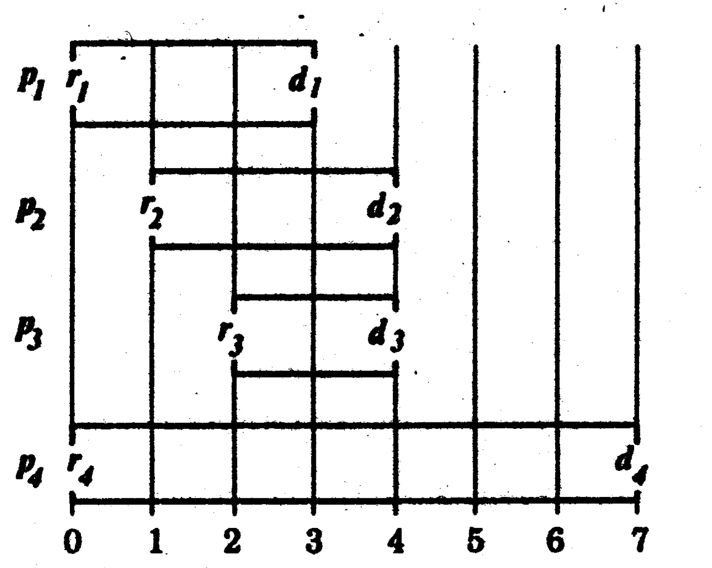
Preemptive scheduling: planing game(2, t=1)



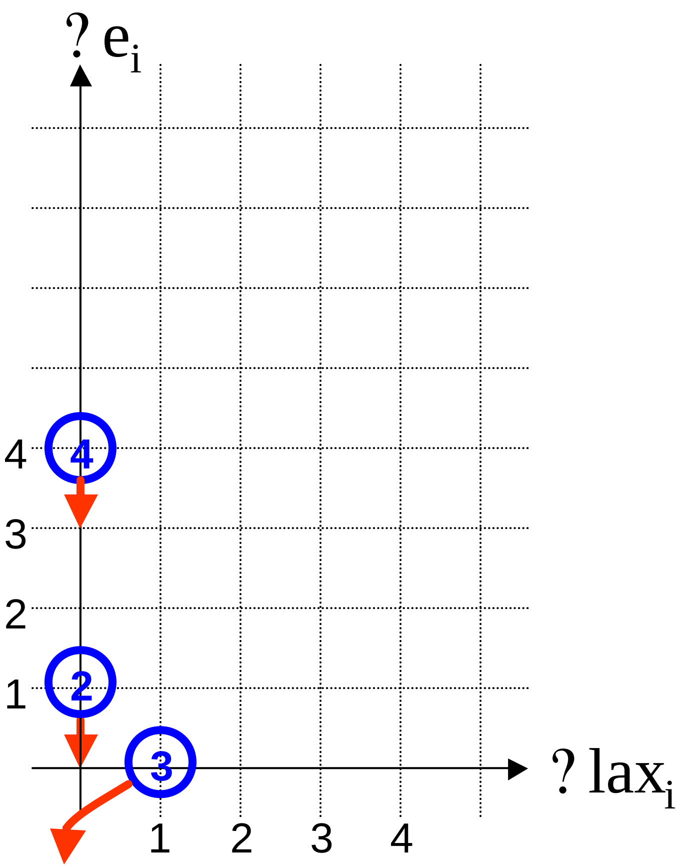


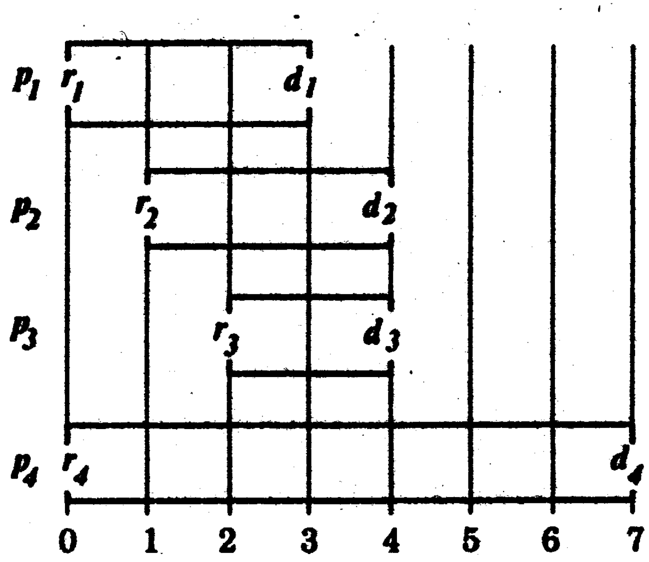
Preemptive scheduling: planing game(2, t=2)



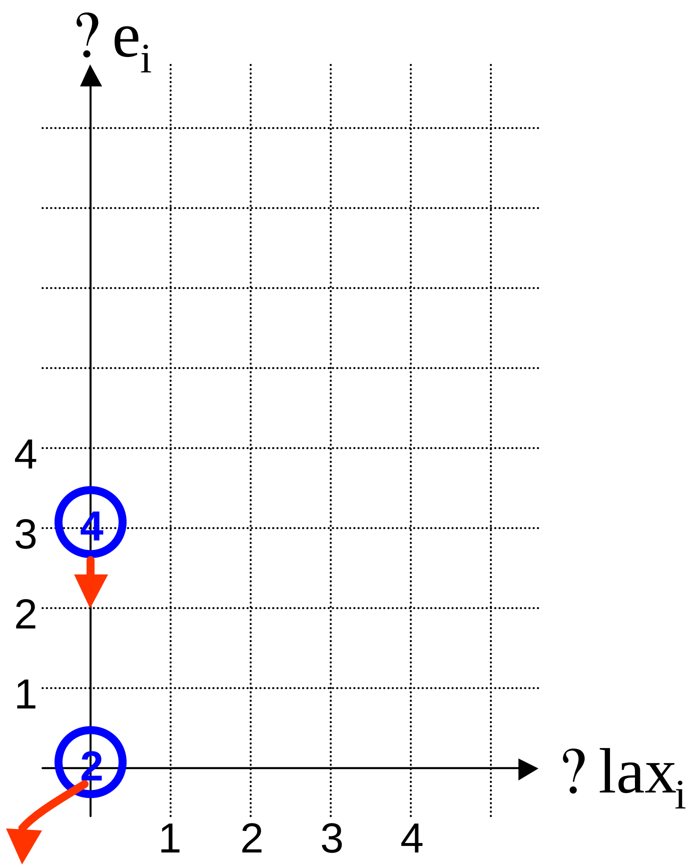


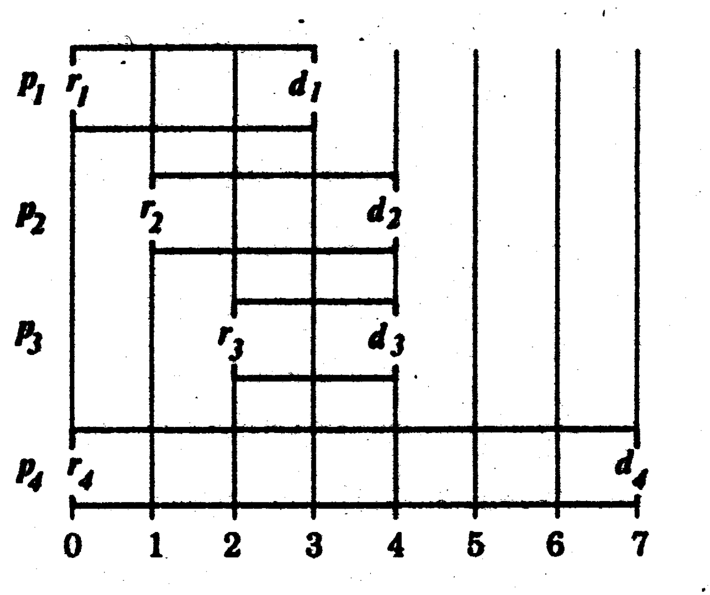
Preemptive scheduling: planing game(2, t=3)



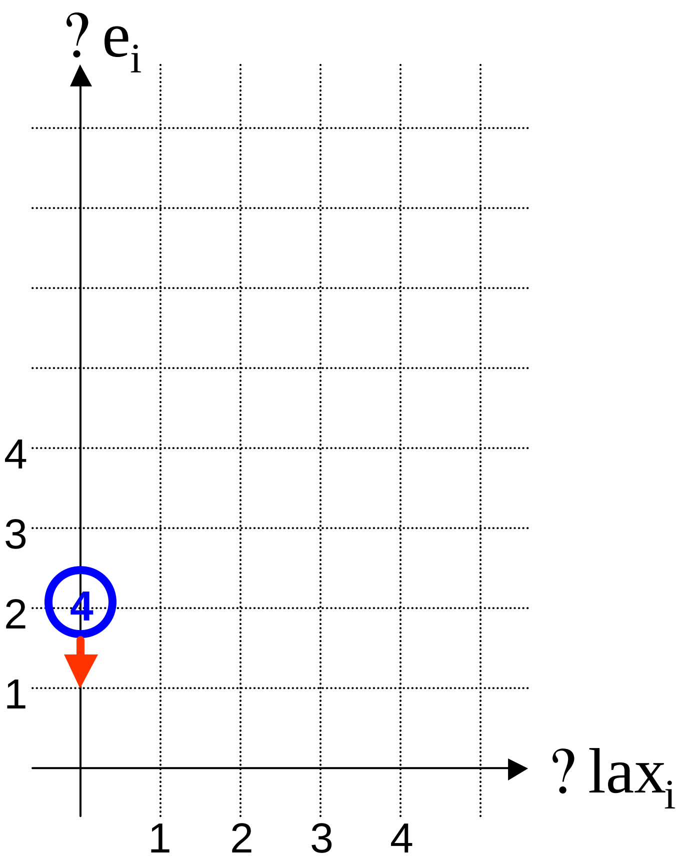


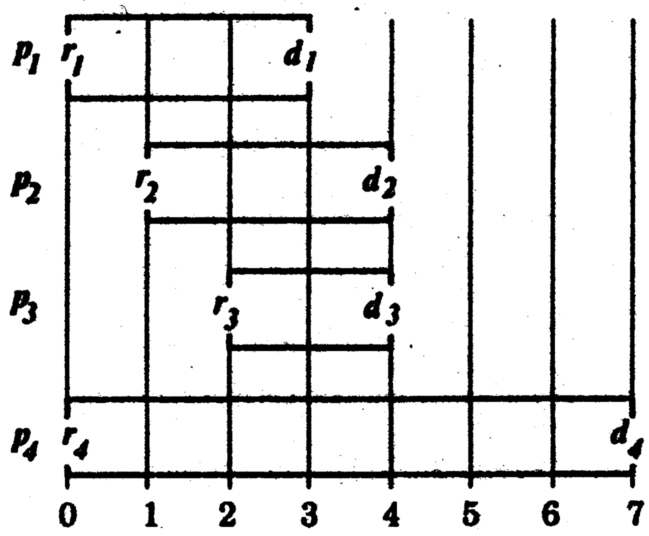
Preemptive scheduling: planing game(2, t=4)



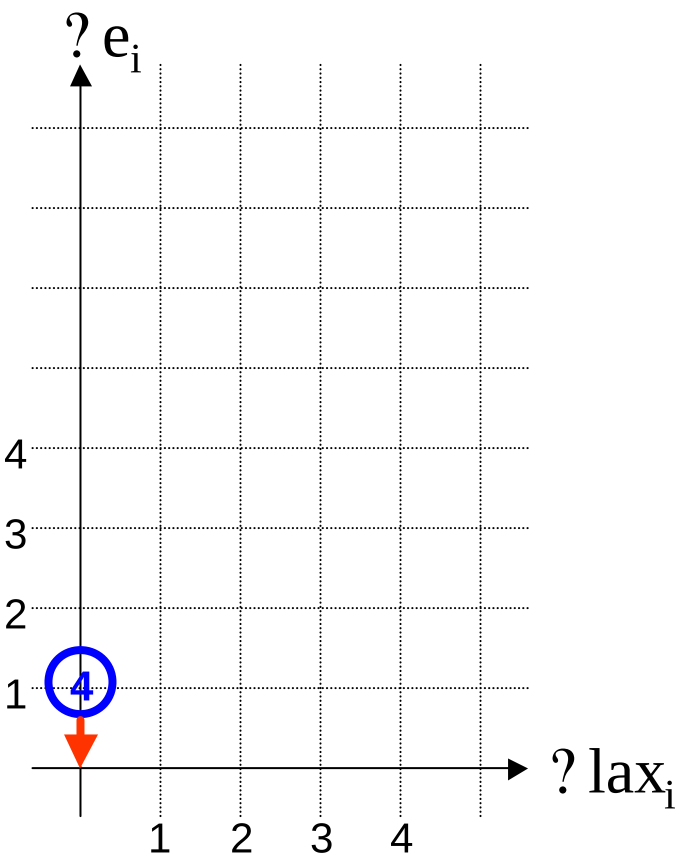


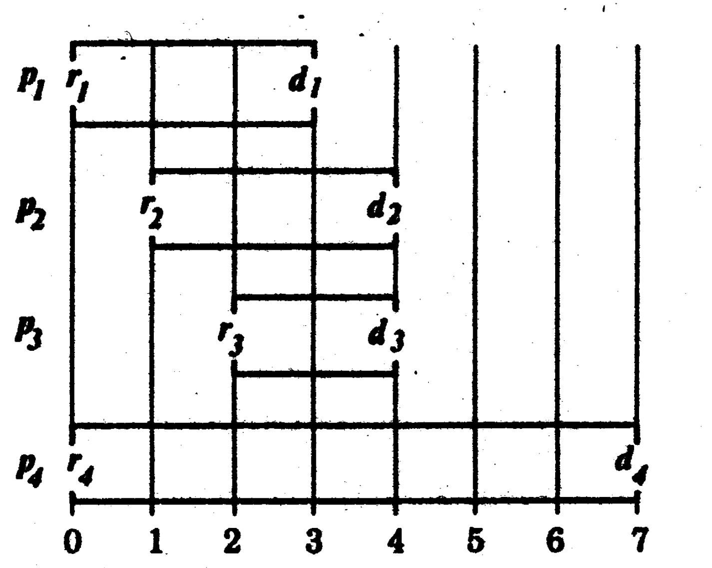
Preemptive scheduling: planing game(2, t=5)



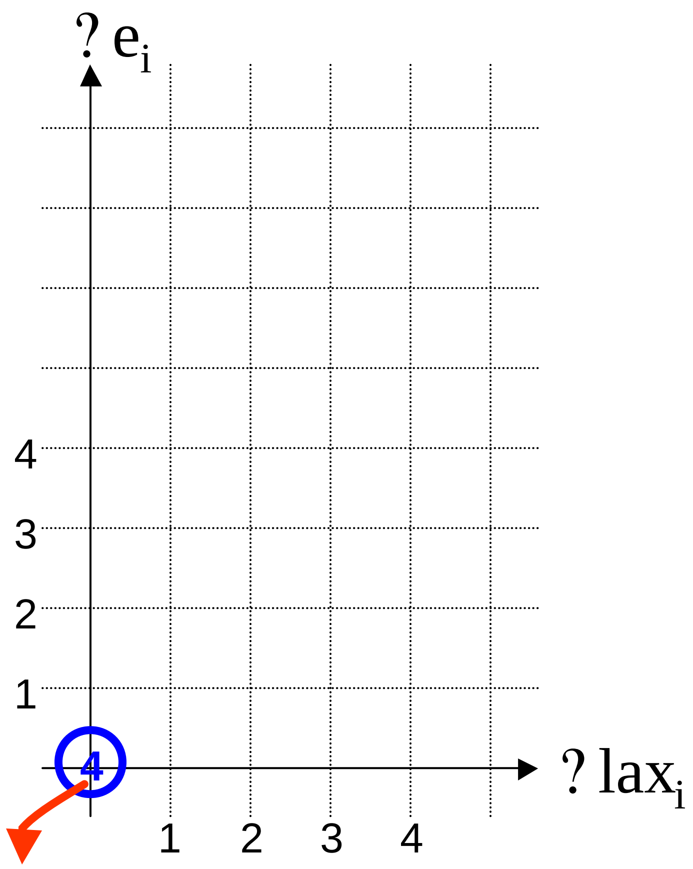


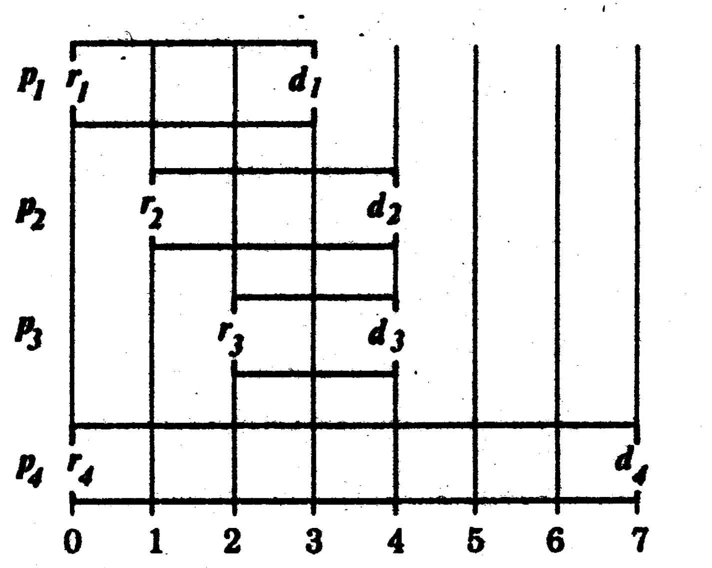
Preemptive scheduling: planing game(2, t=6)

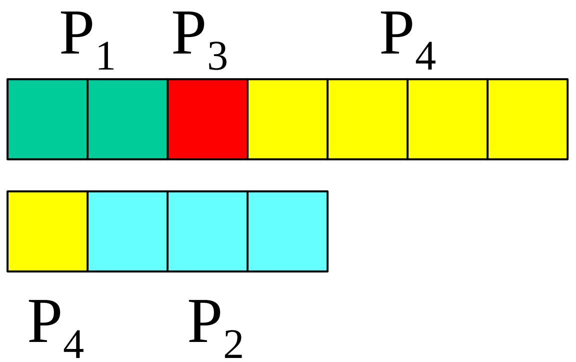




Preemptive scheduling: planing game(2, t=7)







Comparison scheduling strategies

Strategie des Planungverfahrens	angewandt auf unterbrechbare Prozesse	angewandt auf nicht unterbrechbare Prozesse
Planen durch Suchen		Optimale Plane bei einem Aufwand von $O(n!)$, grundsätzlich: Problem ist NP-vollständig
Planen nach Fristen	Optimal für sporadische und periodische Prozesse, für statische und dynamische Planungsverfahren	Optimal bei Prozessen mit gleichen Bereitzeiten
Planen nach Spielräumen	Bei Mehrprozessorsystemen: optimal für statische Planungsverfahren, nicht optimal bei dynamischen Planungsverfahren	Bei den Mehrprozessorsystemen unabhängig von der Strategie des Planungsverfahrens: NP-vollständig schon bei 2 Prozessoren mit gleichen Bereitzeiten und gleichen Fristen
Planen nach monotonen Raten	Für n periodische Prozesse: optimal nur für Auslastungen $U \leq n(\sqrt[n]{2} - 1)$, dafür aber mittels Prioritätszuordnung einfach implementierbar	