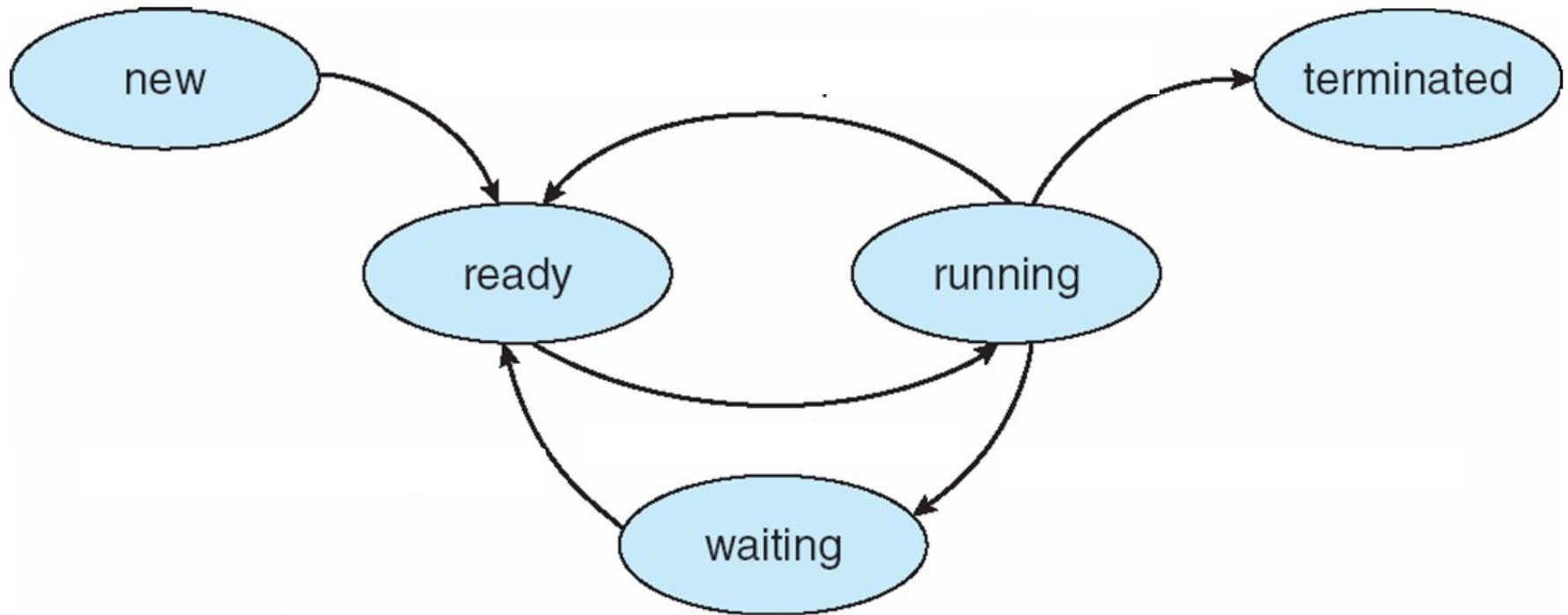


Планиране на процесора

проф. д-р инж. Христо Вълчанов

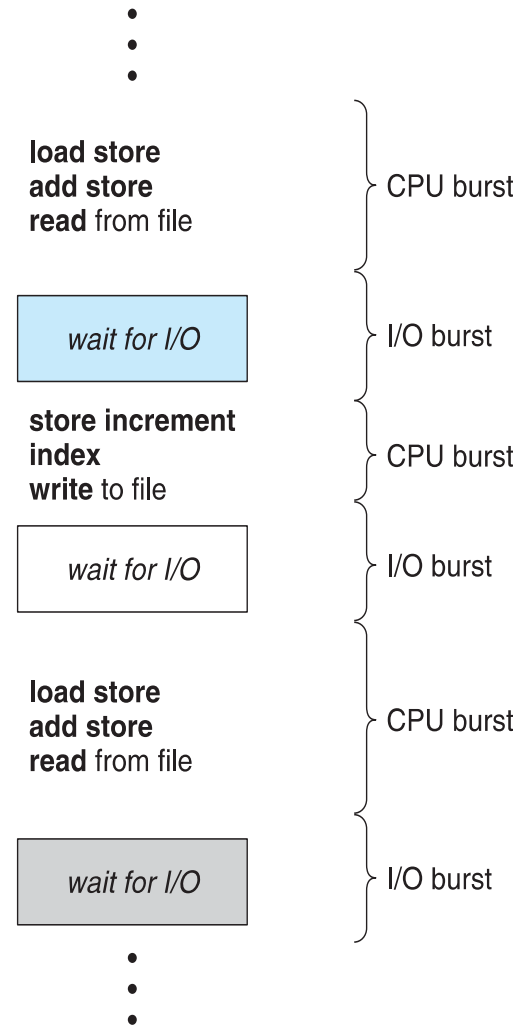
<http://cs.tu-varna.bg>

Състояния на процесите



Основни концепции

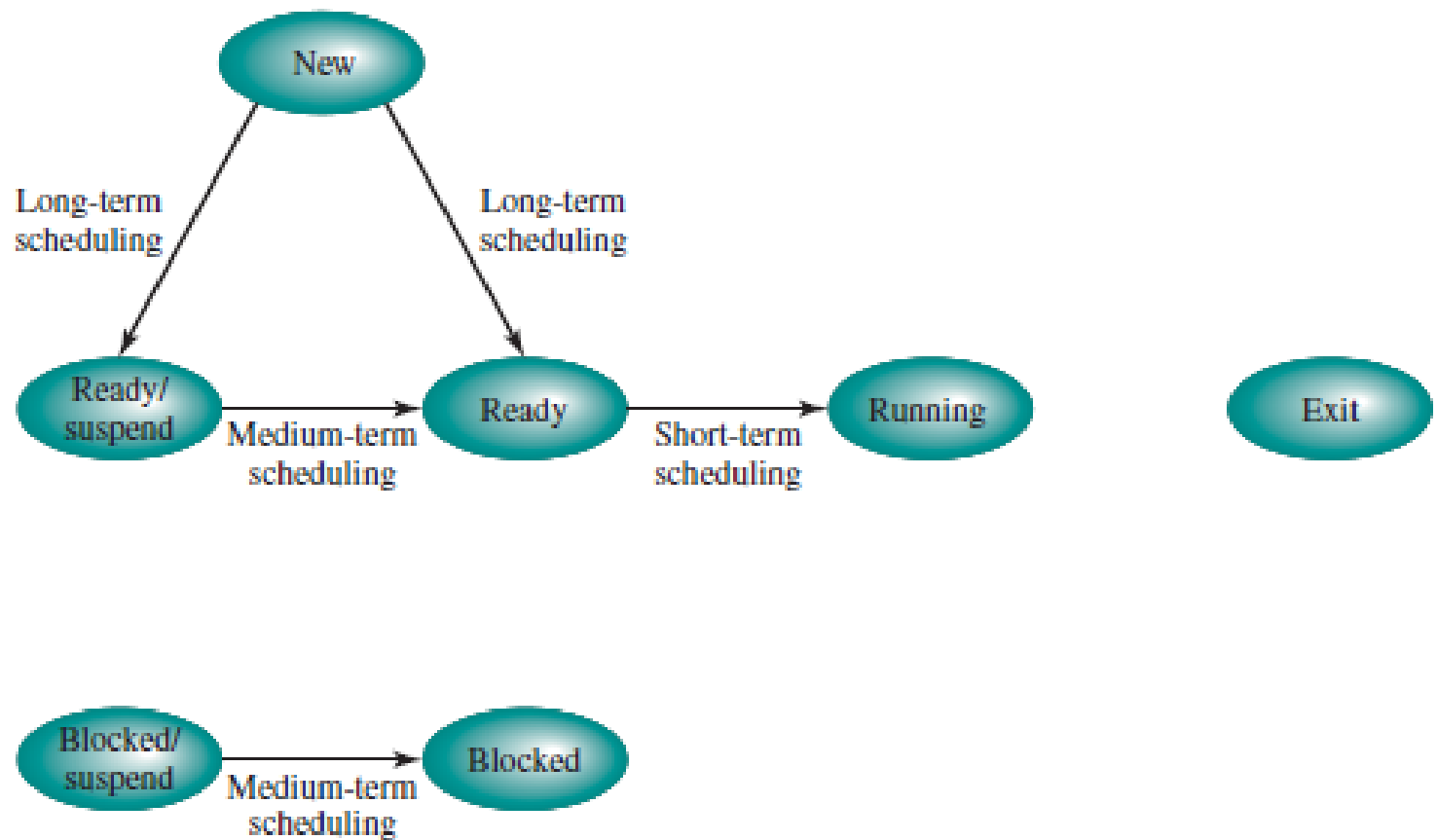
- Максимално използване на CPU се получава чрез мултипрограмно изпълнение
- CPU–I/O Burst Cycle – изпълнението на процес включва цикли работа на CPU и I/O изчаквания
- **CPU burst** следван от **I/O burst**
- CPU burst разпределянето е от голямо значение



Видове планирания

- **Short-term scheduling** – кой наличен процес ще се изпълни върху процесора;
- **Medium-term scheduling** – кои процеси да се добавят към готовите в паметта;
- **Long-term scheduling** – при създаване на нов процес;
- **I/O scheduling** – за кой процес I/O заявки да се обслужат от налично I/O устройство.

Видове планирания (2)



CPU планировчик (Scheduler)

- **Short-term scheduler** избира един от процесите *от Ready Queue* и заема за него CPU
 - Опашката може да бъде подредена по различни начини
- Кога се изисква планиране на CPU – ако процес:
 1. Се превключва от Running в Waiting състояние (заявка за I/O)
 2. Се превключва от Running в Ready (прекъсване)
 3. Се превключва от Waiting в Ready (завършване I/O)
 4. Завършва

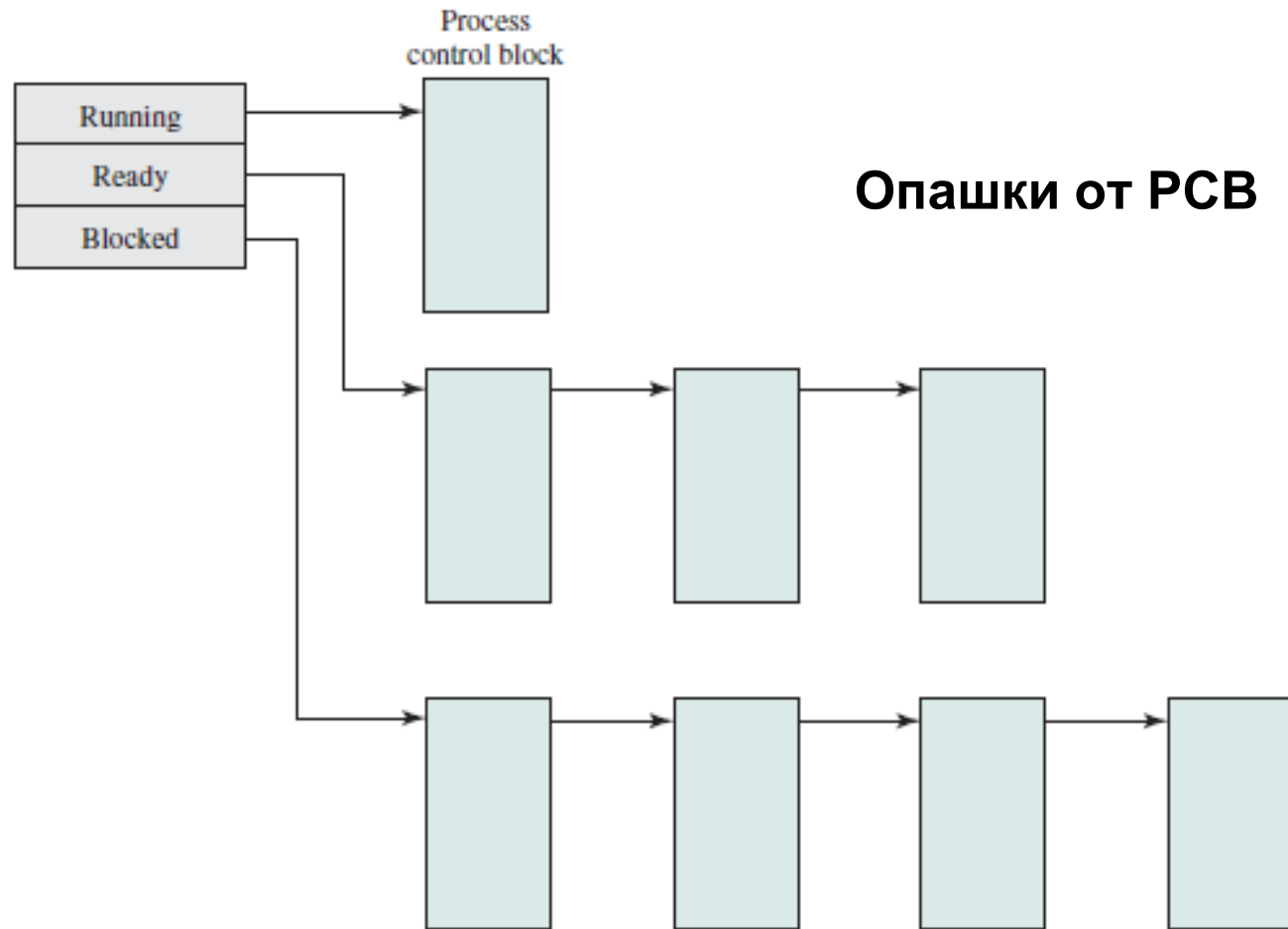
Preemptive и non-preemptive планиране

- **Non-preemptive** – след като CPU е зает от процес, процесът притежава CPU, докато го освободи или при завършване или при преминаване в waiting състояние
- **Preemptive** -- CPU може да бъде даден на друг процес по всяко време. Проблемни ситуации:
 - Достъп до споделени данни
 - Изпълнение в kernel mode
 - Настъпване на прекъсвания по време на критични действия на ОС

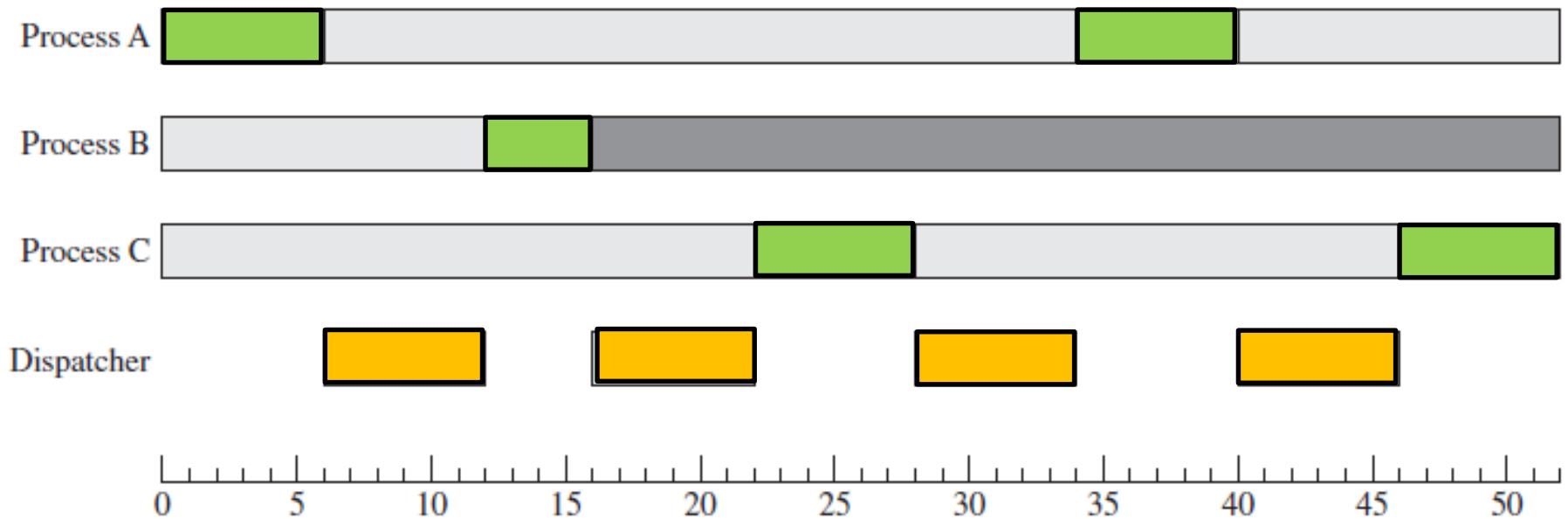
Диспечер

- Диспечер е софтуерен модул, предоставящ контрола върху CPU на избраните за изпълнение процеси. Това включва:
 - Превключване на контекста
 - Превключване в потребителски режим
 - Преход към подходящ адрес в потребителската програма за възстановяване на нейното изпълнение
- **Dispatch latency** – времето, необходимо на диспечера да спре един процес и да стартира друг

Организация на състоянията на процесите



Изпълнение на процесите



Критерии за планиране

- **Използване на CPU** – съхраняване CPU натоварен колкото се може повече време
- **Throughput** – брой процеси, завършили своето изпълнение за единица време
- **Turnaround time** – време за изпълнение на отделен процес
- **Waiting time** – време за чакане на процес в ready queue
- **Response time** – време от момента на изпратената заявка до генерирането на първият отговор в интерактивните системи

Алгоритъм на планиране

- Оптимизационни критерии
 - Max CPU заемане
 - Max throughput
 - Min turnaround time
 - Min waiting time
 - Min response time
- Ще се разглежда за всеки процес случай на единичен CPU burst
- **Gantt chart** - бар-диаграма, описваща частично планиране с начално и крайно време за процес

$$\text{WaitingTime} = \text{ServiceTime} - \text{ArrivalTime}$$

First- Come, First-Served (FCFS) планиране

<u>Process</u>	<u>Burst Time</u>
P_1	24
P_2	3
P_3	3

- Нека процесите са в последователност: P_1 , P_2 , P_3 във време 0.



- Waiting time за $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- Средно waiting time: $(0 + 24 + 27)/3 = 17$

FCFS планиране

Ако последователността на процесите:

$$P_2, P_3, P_1$$



- Waiting time for $P_1 = 6$; $P_2 = 0$; $P_3 = 3$
- Средно waiting time: $(6 + 0 + 3)/3 = 3$
- По-добър резултат
- **Convoy effect** – късите процеси са зад по-дългите

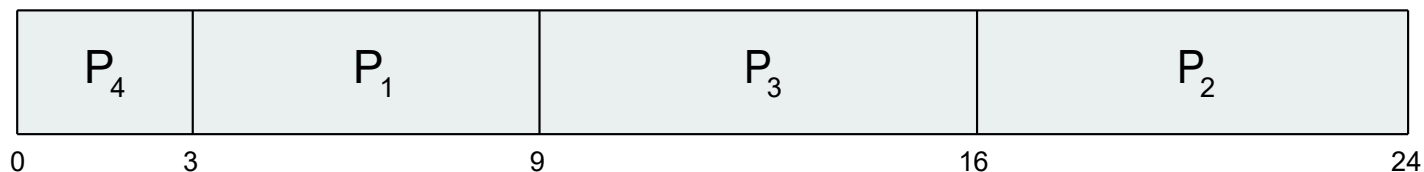
Shortest-Job-First (SJF) планиране

- С всеки процес се асоциира продължителността на неговия следващ CPU burst
 - Тези продължителности се използват за планиране на по-късите процеси
 - SJF е оптимален – дава minimum средно време за изчакване за дадено множество от процеси
 - Трудността е определянето на продължителността на следващата CPU заявка
-
- Лесен за реализация в системи с пакетна обработка.
 - Не може да се реализира в интерактивни системи.

Пример за SJF

<u>Process</u>	<u>Burst Time</u>
P_1	6
P_2	8
P_3	7
P_4	3

- SJF диаграма



- Средно waiting time = $(3 + 16 + 9 + 0) / 4 = 7$

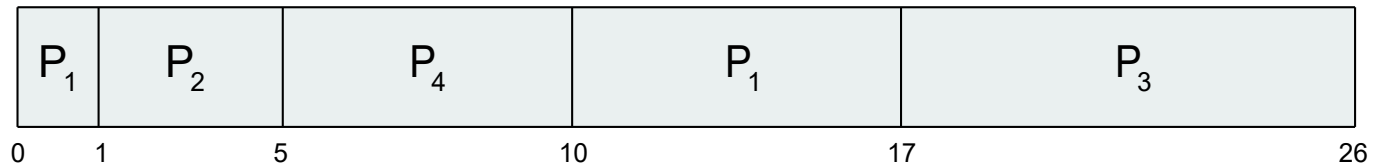
Shortest-remaining-time-first планиране

- Preemptive версия на SJF.
 - Процесорът се предоставя на задача с оставащо за изпълнение най-малко време, но може да изтласкан от нова задача с по-малко време на завършване.
-
- Използва се в системи с пакетна обработка.
 - Не може да се реализира в интерактивни системи.

Shortest-remaining-time-first планиране

- Preemptive версия на SJF се нарича **shortest-remaining-time-first**

<u>Process</u>	<u>Arrival Time in ReadyQ</u>	<u>Burst Time</u>
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5



- Средно waiting time = $[(10-1)+(1-1)+(17-2)+(5-3)]/4 = 26/4 = 6.5$

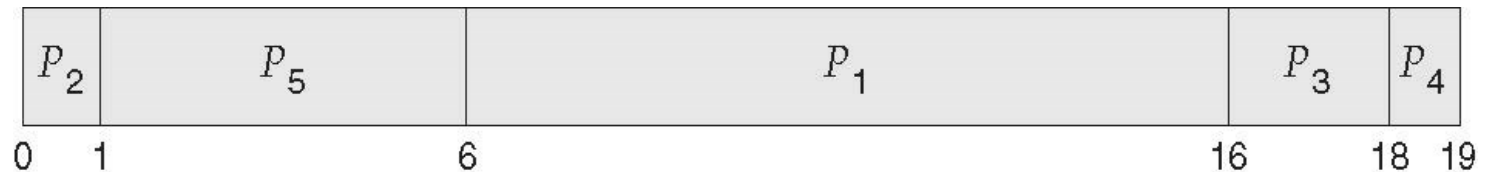
Приоритетно планиране

- Приоритет – цяло число, асоциирано с всеки процес
- CPU се заема от процеса с най-висок приоритет (по-малко число = по-висок приоритет)
 - Preemptive
 - Non-preemptive
- Проблем \equiv **Starvation** – ниско-приоритетните процеси могат никога да не се изпълнят
- Решение \equiv **Aging** – приоритетът да нараства за процеси, чакащи дълго време

Пример

Процесите се появяват във време 0

<u>Process</u>	<u>Burst Time</u>	<u>Priority</u>
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2



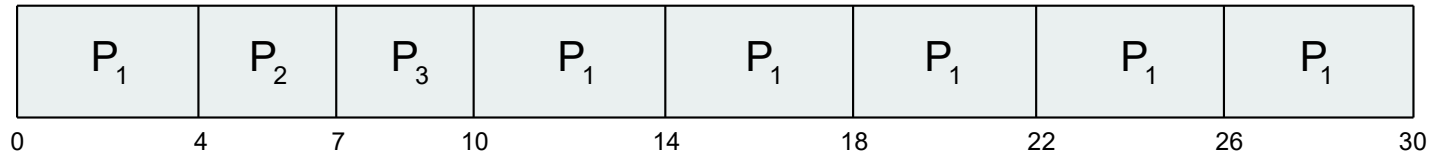
- Средно waiting time = $(0 + 1 + 6 + 16 + 18)/5 = 8.2$

Round Robin (RR)

- Всеки процес използва CPU кратко време (**time quantum q**). След изтичането на това време, процесът се изтласква и се добавя в края на ready queue.
- За планиране се използват прекъсвания от таймер
- Производителност
 - q е голямо \Rightarrow FIFO
 - q е малко \Rightarrow превключването води до големи разходи

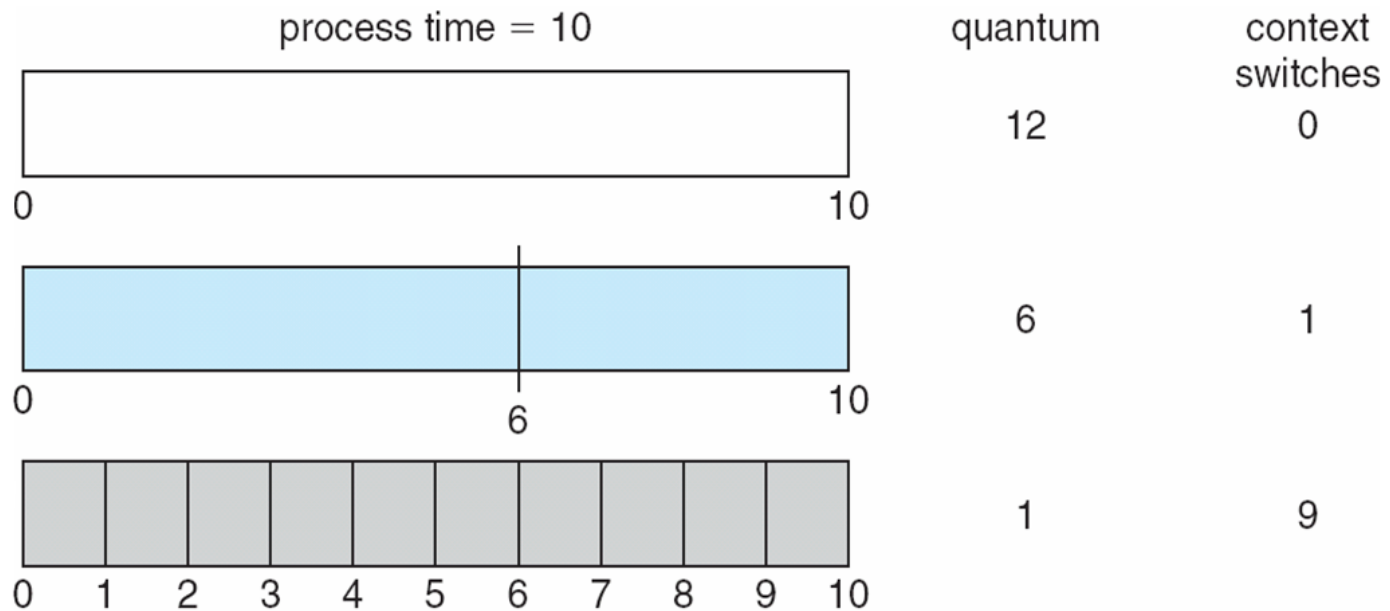
Пример RR с квант = 4

<u>Process</u>	<u>Burst Time</u>
P_1	24
P_2	3
P_3	3



- Типично, RR има по-голям среден **turnaround** от SJF, но по-добър **response**
- q трябва да е голямо в сравнение с времето за превключване на контекста
- q обикновено е от 10ms до 100ms, превключването на контекста < 10 usec

Кванти и време за превключване на контекста

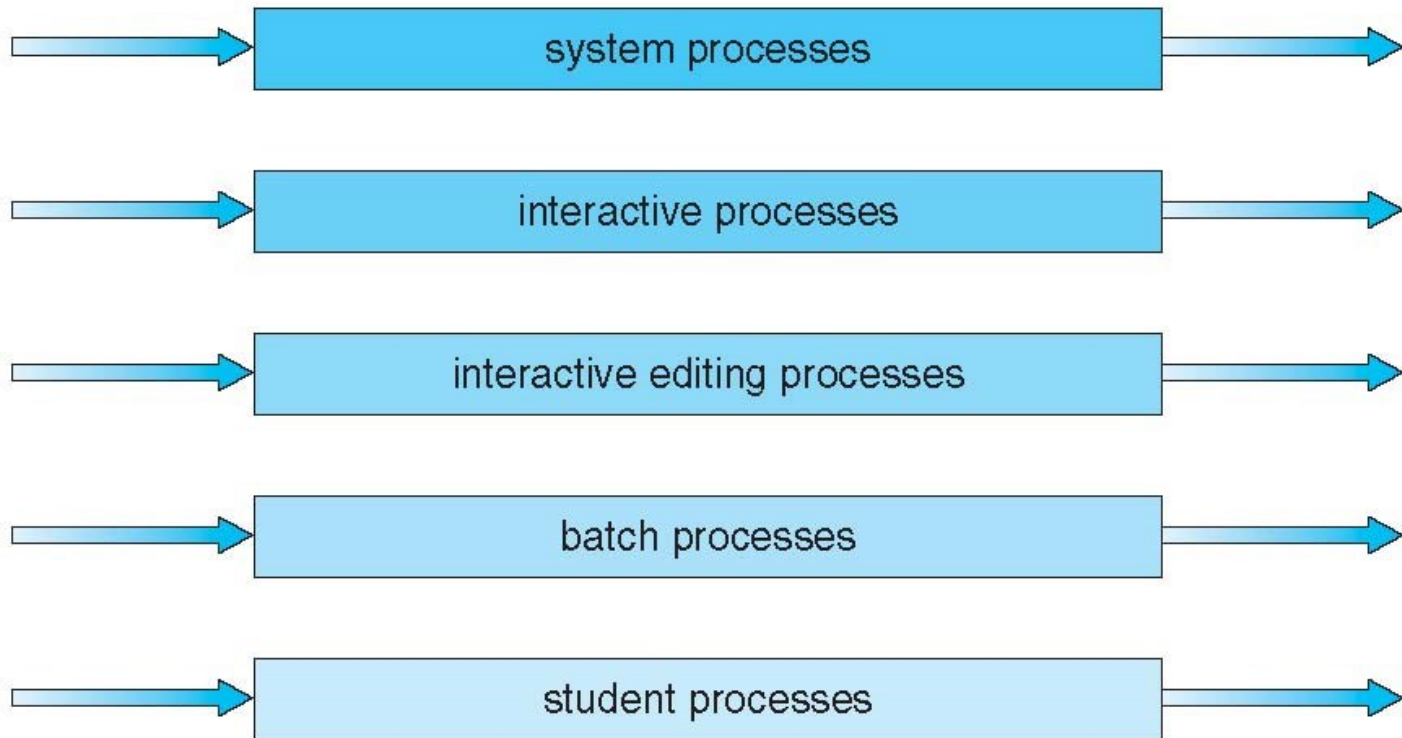


Multilevel Queue

- Ready queue се разделя на няколко опашки според вида процеси, например:
 - **foreground** (interactive)
 - **background** (batch)
- Всяка опашка има собствен алгоритъм на планиране:
 - foreground – RR
 - background – FCFS
- Трябва да се планира между опашките:
 - **Fixed priority scheduling** – обслужват се първо всички процеси от foreground опашката, после от background опашката. Възможна е starvation.
 - **Time slice** – всяка опашка получава определена продължителност от CPU времето, която да планира между процесите си, например 80% за foreground при RR и 20% за background при FCFS

Multilevel Queue планиране

highest priority



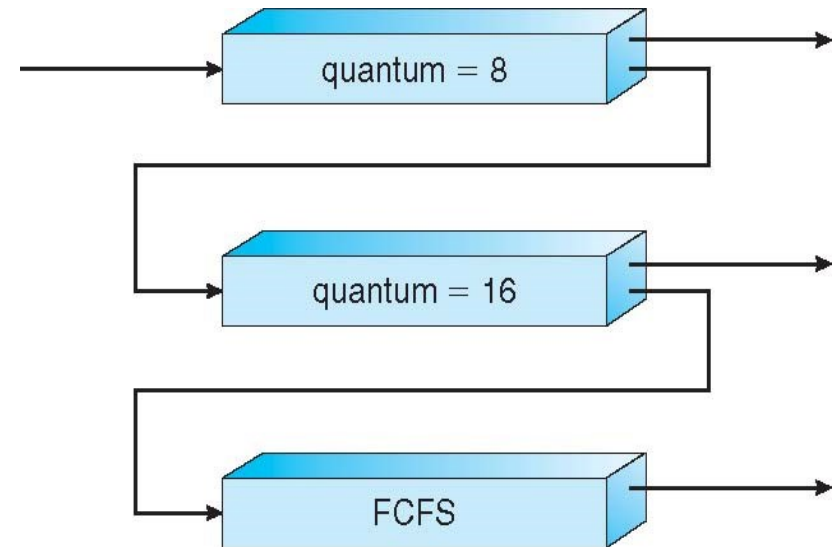
lowest priority

Multilevel Feedback Queue

- Множество опашки
- Процес може да бъде преместван между различни опашки
- Планирането се базира на :
 - Брой опашки
 - Използвания метод за определяне в коя опашка процес ще постъпи при необходимост от обслужване
 - Всяка опашка има отделен алгоритъм на планиране
 - Използвания метод за определяне кога да се постави процес в по-приоритетната опашка
 - Използвания метод за определяне кога да се намали приоритета на процес
- Aging може да се реализира чрез увеличаване приоритета на процес

Пример на Multilevel Feedback Queue

- Три опашки:
 - Q_0 – RR с квант 8 ms
 - Q_1 – RR с квант 16 ms
 - Q_2 – FCFS
- Планиране
 - Нова задача влиза в опашка Q_0 , обслужвана по FCFS
 - Когато заеме CPU, процесът получава 8 ms
 - Ако не завърши в рамките на 8ms, процесът се премества в Q_1
 - В Q_1 процесът отново се обслужва по FCFS и получава допълнителни 16ms
 - Ако все още не завърши, се изтласква и се премества в Q_2



Планиране в Linux

Приоритети:

- Статичен
 - За real-time процеси
 - Диапазон 0 - 99
- Динамичен
 - За конвенционални процеси
 - Диапазон 100-139
 - Променя се в зависимост от изпълнението на процеса

Приоритет на процес

- Базиран на “*nice*” ниво на процеса.
 - По подразбиране 0, диапазон -20 to +19 (по-малкото е по-добро)
- I/O bound процеси получават по-висок приоритет.
- CPU bound процеси получават малък приоритет.

Приоритет на процес

194.141.24.72 - PuTTY

Tasks: 138 total, 1 running, 137 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.2 us, 0.2 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 8060020 total, 2138812 free, 5134688 used, 786520 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 2537864 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
5024	root	20	0	20540	2928	2456	R	1.0	0.0	0:00.13	top
7	root	20	0	0	0	0	S	0.7	0.0	99:37.46	rcu_sched
3	root	20	0	0	0	0	S	0.3	0.0	0:50.13	ksoftirqd/0
1	root	20	0	4372	1436	1332	S	0.0	0.0	0:17.44	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.76	kthreadd
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:+
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0.0	0.0	0:00.07	migration/0
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.68	migration/1
11	root	20	0	0	0	0	S	0.0	0.0	0:00.77	ksoftirqd/1
13	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/1:+
14	root	20	0	0	0	0	S	0.0	0.0	0:00.03	kdevtmpfs
15	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	netns
16	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	perf
17	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	writeback
18	root	39	19	0	0	0	S	0.0	0.0	0:20.61	khugepaged
19	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	crypto
20	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kintegrityd
21	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	bioset
22	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kblockd
23	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	ata_sff
24	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	md
25	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	devfreq_wg

Приоритет и Nice

Nice = -20 .. +19

PR = 20 + Nice

nice -n nice_value command

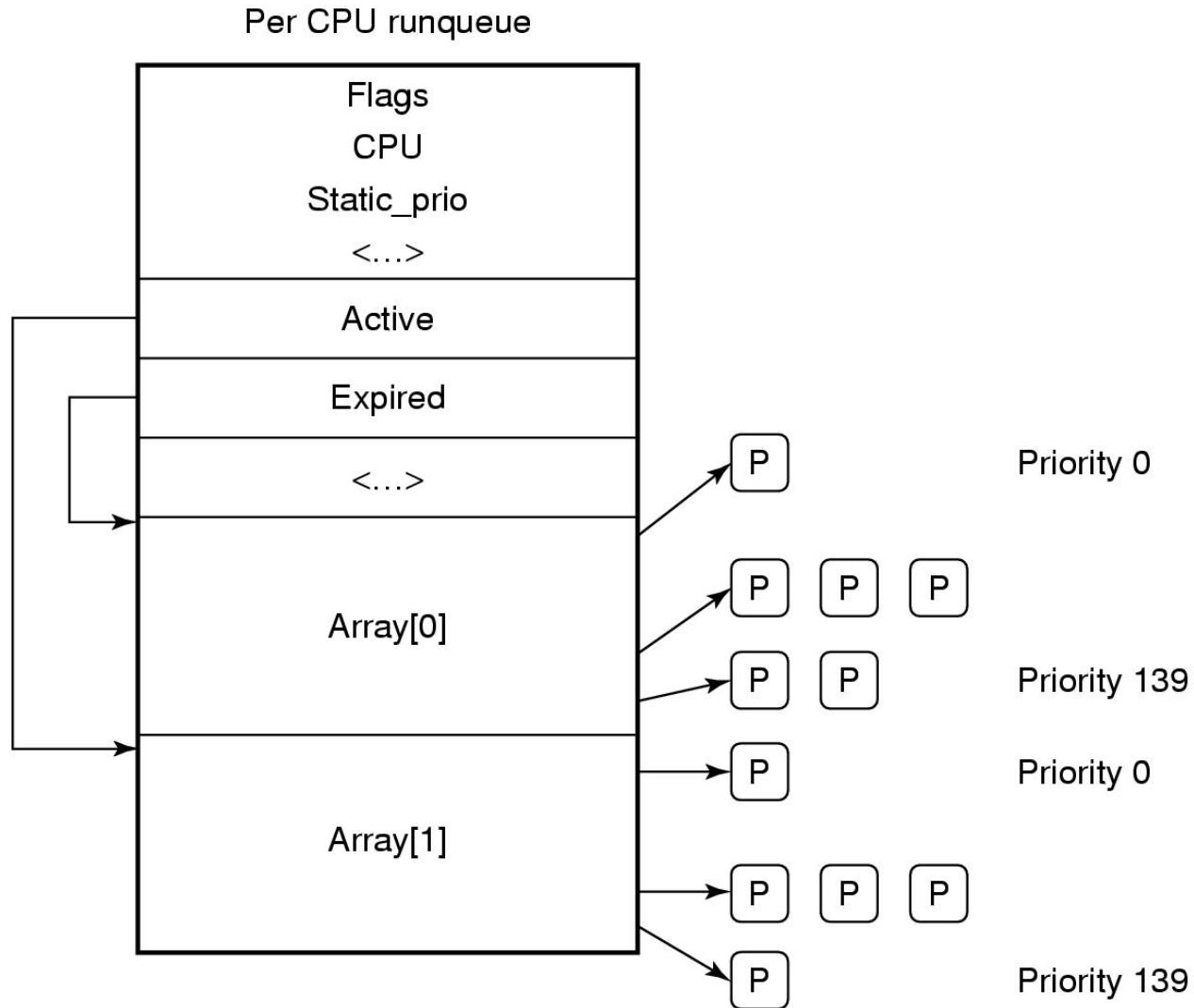
Квант на процес

- При създаване child получава $\frac{1}{2}$ от timeslice на родителя
- Високия приоритет получава по-голям timeslice
 - Нисък приоритет/ по-малко интерактивен
 - Висок приоритет/ повече интерактивен
- Когато timeslice $\rightarrow 0$, процесът става *expired*

Квант за конвенционални процеси

Description	Static priority	Base Time quantum (timeslice)
Highest static priority	100	800 ms
High static priority	110	600 ms
Default static priority	120	100 ms
Low static priority	130	50 ms
Lowest static priority	139	5 ms

Опашки за изпълнение



Планиране в Windows

- Базирано е на приоритети
- Preemptive
- За всяка нишка се назначава квант от време

Processor Affinity

Affinity – конфигуриране кои процесори могат да се използват от процес

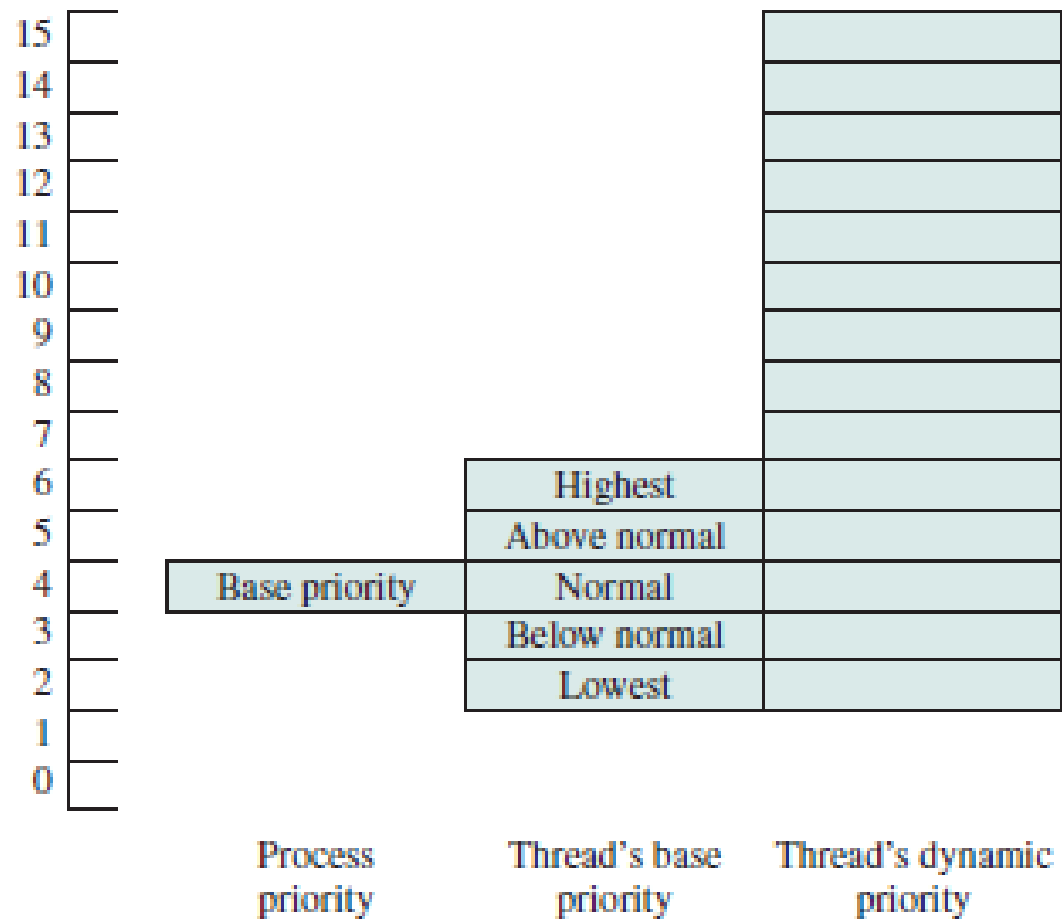
Планиране в Windows

Нива от 0 до 31

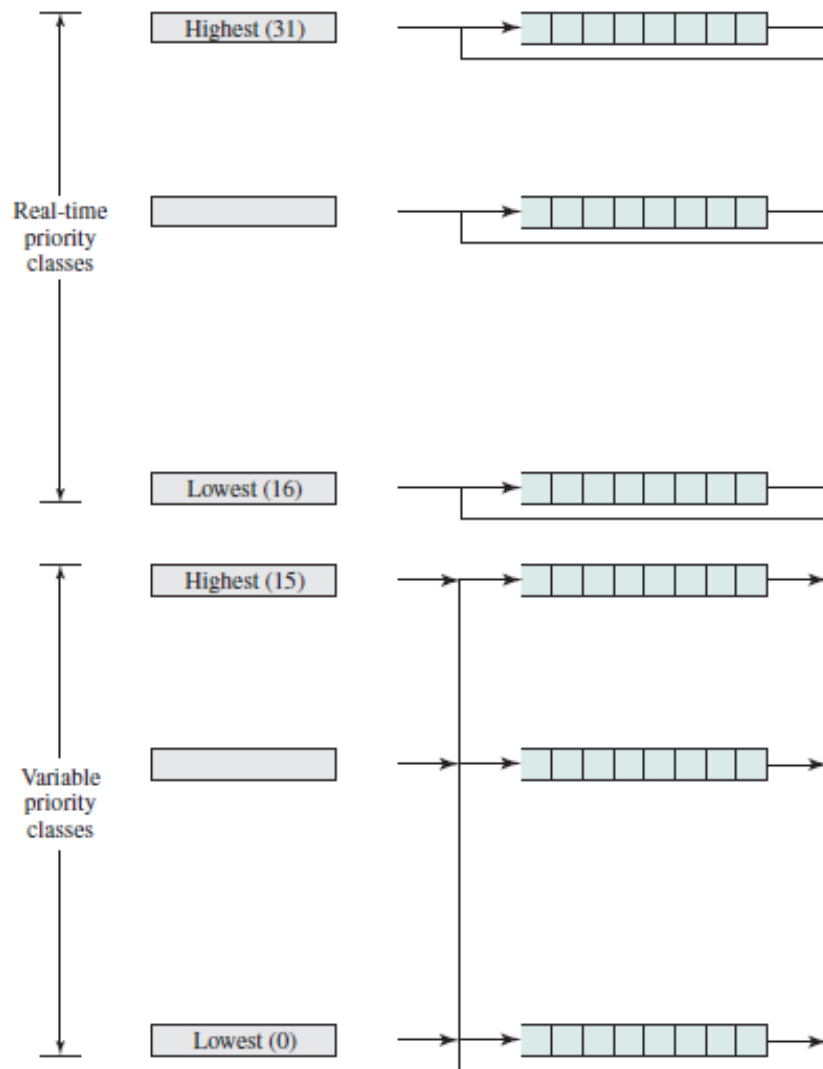
Два класа приоритети:

- Real-time (16-31)
- Variable priority (1-15)

Приоритети в Windows



Приоритети в Windows



Планиране в Windows

Task Manager

File Options View

Processes Performance App history Startup Users Details Services

Name	PID	Status	User name	CPU	Working set (memory)	Page faults	Base priority	Description
ApplicationFrameHo...	4944	Running	hwalchan	00	4 740 K	20 036	Normal	Application Frame Host
armsvc.exe	1620	Running	SYSTEM	00	880 K	1 971	Normal	Adobe Acrobat Update Service
cmEvtSrv64.exe	1892	Running	SYSTEM	00	584 K	983	Normal	charismathics smart security service
csrss.exe	664	Running	SYSTEM	00	2 160 K	5 761	Normal	Client Server Runtime Process
csrss.exe	788	Running	SYSTEM	00	3 020 K	1 037 824	Normal	Client Server Runtime Process
ctfmon.exe	4164	Running	hwalchan	00	8 948 K	24 177	High	CTF Loader
dasHost.exe	7704	Running	LOCAL SERVICE	00	936 K	1 473	Normal	Device Association Framework Provid
dllhost.exe	5656	Running	hwalchan	00	2 592 K	4 178	Normal	COM Surrogate
dwm.exe	1268	Running	DWM-1	00	59 076 K	227 891 882	High	Desktop Window Manager
explorer.exe	4504	Running	hwalchan	00	139 640 K	1 286 909	Normal	Windows Explorer
firefox.exe	2706	Running	hwalchan	00	623 156 K	1 000 499	Normal	Firefox
firefox.exe			hwalchan	00	89 152 K	476 712	Normal	Firefox
firefox.exe			hwalchan	00	125 232 K	428 213	Low	Firefox
firefox.exe				00	106 044 K	120 017	Normal	Firefox
firefox.exe				00	118 228 K	110 350	Normal	Firefox
firefox.exe				00	201 544 K	2 292 658	Normal	Firefox
firefox.exe				00	70 712 K	55 913	Normal	Firefox
firefox.exe				00	40 580 K	11 540	Normal	Firefox
FMAPP.exe				00	8 436 K	2 237	Normal	FMAPP Application
fontdrvho			UMFD-0	00	1 564 K	1 810	Normal	Usermode Font Driver Host
fontdrvho			UMFD-1	00	7 712 K	34 478	Normal	Usermode Font Driver Host
hpqbam			hwalchan	00	6 772 K	4 440	Normal	HP CUE Alert Popup Window Objects

End task
End process tree
Set priority
Set affinity
Analyze wait chain
Debug
UAC virtualization
Create dump file
Open file location
Search online
Properties
Go to service(s)

Realtime
High
Above normal
Normal
Below normal
Low

Fewer details

Операционни системи: Христо Вълчанов

Планиране в Windows

Task Manager

File Options View

Processes Performance App history Startup Users Details Services

Name	PID	Status	User name	CPU	Working set (memory)	Page faults	Base priority	Description
ApplicationFrameHo...	4944	Running	hwalchan	00	4 740 K	20 036	Normal	Application Frame
armsvc.exe	16			0	880 K	1 971	Normal	Adobe Acrobat U
cmEvtSrv64.exe	18			0	584 K	983	Normal	charismathics sma
csrss.exe	66			0	2 144 K	5 765	Normal	Client Server Runti
csrss.exe	78			0	3 000 K	1 048 341	Normal	Client Server Runti
ctfmon.exe	41			0	8 964 K	24 316	High	CTF Loader
dashHost.exe	77			0	936 K	1 473	Normal	Device Association
dllhost.exe	56			0	2 592 K	4 178	Normal	COM Surrogate
dwm.exe	12			0	64 492 K	227 957 213	High	Desktop Window I
explorer.exe	45			0	137 236 K	1 295 235	Normal	Windows Explorer
firefox.exe	97			9	58 992 K	26 788	Low	Firefox
firefox.exe	37			0	425 136 K	1 061 735	Normal	Firefox
firefox.exe	44			0	89 220 K	492 678	Normal	Firefox
firefox.exe	29			0	126 028 K	429 477	Low	Firefox
firefox.exe	70			0	109 324 K	121 455	Normal	Firefox
firefox.exe	47			0	113 608 K	123 896	Normal	Firefox
firefox.exe	10			0	199 108 K	2 356 257	Low	Firefox
FMAPP.exe	7048	Running	hwalchan	00	8 436 K	2 237	Normal	FMAPP Applicatio
fontdrvhost.exe	608	Running	UMFD-0	00	1 564 K	1 810	Normal	Usermode Font Dr
fontdrvhost.exe	576	Running	UMFD-1	00	8 416 K	34 846	Normal	Usermode Font Dr
hpqbam08.exe	4480	Running	hwalchan	00	6 772 K	4 440	Normal	HP CUE Alert Popu
hpqapc01.exe	8516	Running	hwalchan	00	7 852 K	7 237	Normal	GPCore COM obje

Processor affinity

Which processors are allowed to run "firefox.exe"?

- ☒ <All Processors>
- ☒ CPU 0
- ☒ CPU 1
- ☒ CPU 2
- ☒ CPU 3

OK Cancel

^ Fewer details

Планиране в Windows

- Windows 3.1 – non-preemptive
- Windows 95 – опростено preemptive
- Windows NT – Multi-level Feedback Queue
- Windows XT – priority preemptive с приоритетни нива и RR за всяко от тях
- Windows 7 – User-mode scheduling
- Windows 8 – WinJS
- Windows 10 – Multi-level Feedback Queue

Въпроси ?