Оперативни Системи

Распоредување процеси

Вон. проф. Д-р Димитар Трајанов
Вон. проф. Д-р Невена Ацковска
Доц. Д-р Боро Јакимовски

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ
И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Цел на предавањето

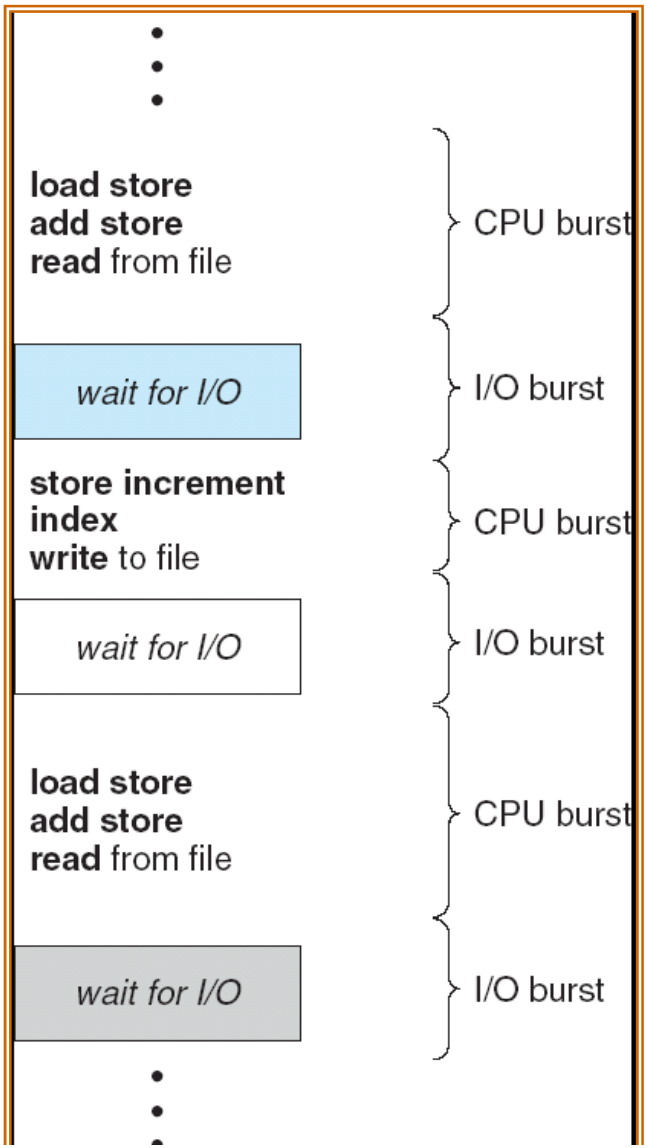
- Распоредување на процеси
- Мерки за перформанси
- Разлики во начините за распоредување
  - Пакетни системи
  - Интерактивни системи
  - Системи кои работат во реално време

Распоредување (Scheduling)

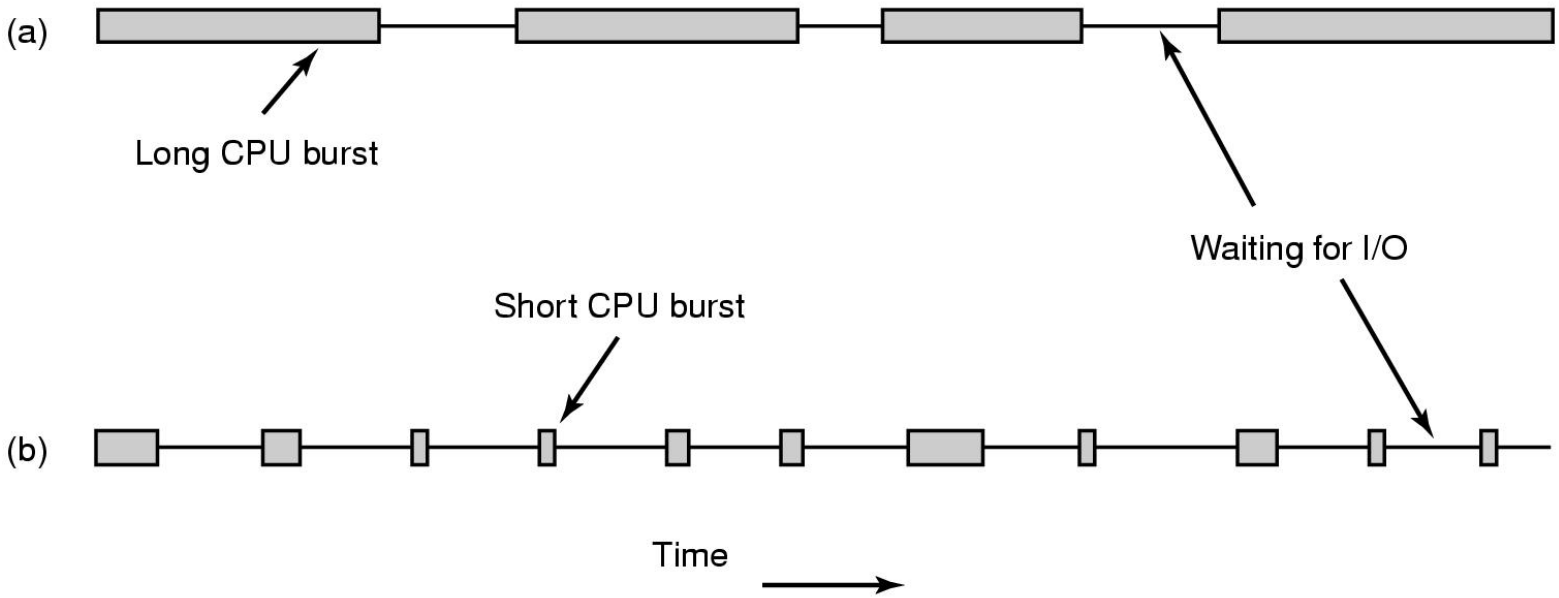
- Кога повеќе од еден процес се во ready состојба треба да се одлучи кој од нив ќе го добие CPU
- Делот од оперативниот систем кој ги прави овие одлуки се нарекува распоредувач (scheduler)
- Распоредувачот работи врз основа на некој алгоритам за распоредување

За секој процес

- извршувањето се состои од циклуси на CPU извршување и I/O чекање



Поделба на процесите



а) CPU доминантен процес
б) I/O доминантен процес

Кога се распоредуваат процеси?

1. Кога процес од извршување доаѓа во состојба на чекање (I/0 барање или повикување wait до завршување на процес дете)
2. Процес оди од состојба на извршување во состојба спремен (се случил прекин)
3. Процес се менува од состојба на чекање во спремна состојба (завршил I/0)
4. Кога терминира процесот

Типови ресурси

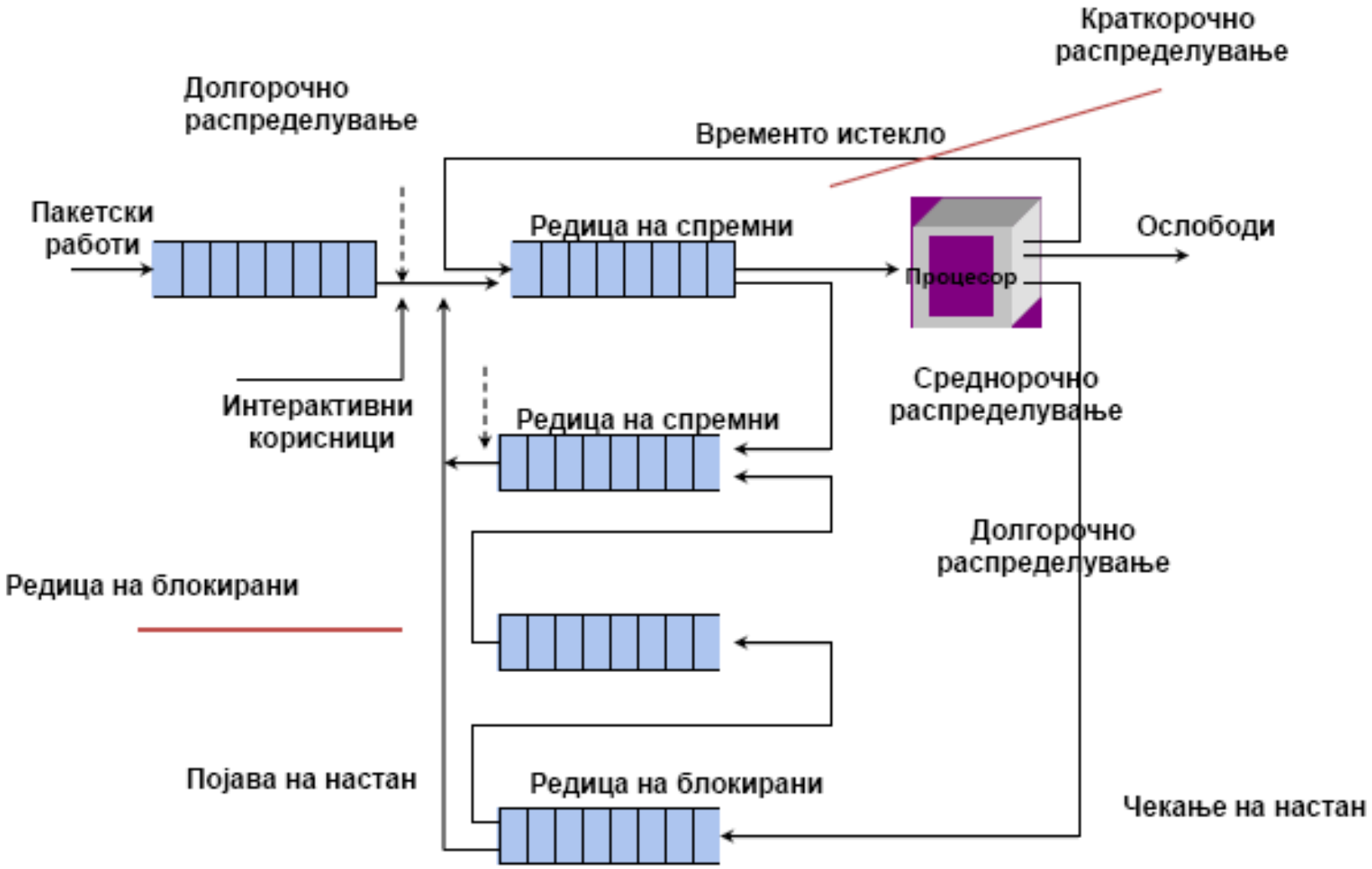
- Ресурсите можат да бидат поделени во една од двете групи
- Типот на ресурси одредува како ОС го управува

1) **Non-preemptible ресурси** (кои не се одземаат)
   - Откако е доделен ресурсот, не може да се користи се' додека не биде ослободен
   - Потребни се повеќе инстанци од ресурсот
   - Пример: Блок на диск
   - ОС менаџмент: алокација
     - Одлучува кој процес го добива кој ресурс
2) **Preemptible ресурси** (кои се одземаат)
   - Може да е одземен ресурсот, потоа се враќа
   - Може да има само еден од овој тип ресурс
   - Пример: CPU
   - ОС менаџмент: распоредување
     - Одлучува по кој редослед се опслужуваат барањата
     - Одлучува колку долго процесот го држи ресурсот

Како се извршува распоредувањето

- Во дадени периодични временски моменти (прекини од системскиот часовник)
  - Non-preemptive: процесот се извршува додека тој самиот не блокира или заврши
  - Preemptive: процесот може да го користи CPU во рамките на некој предефиниран максимален временски период

Дијаграм на редици на распоредување



Мерки за перформанси на распоредувачките алгоритами

- Проток (throughput) –број на процеси завршени во единица време
- Време до завршување, време во систем (turnaround time) – времето поминато од доставувањето до комплетирањето на процесот
- CPU Искористеност – процент од времето во кое процесорот е зафатен
- Време на чекање – време на чекање (за CPU) поминато во редицата на спремни процеси
- Време на одзив – во интерактивни системи времето на завршување не е добра мерка. Време од поднесување барање до прв добиен резултат

Категории на алгоритами за распоредување

- Во зависност од типот на системот постојат различни алгоритами
  - Пакетни (Batch)
    - Пресметка на камати, процесирање на случаи кај осигурителна компанија
    - Нема корисници
  - Интерактивни
    - Preemptive
  - Во реално време (Real-Time)
    - треба да бидат исполнети роковите

Цели на алгоритамите за распоредување (1)

- За сите системи
  - Фер – секој процес да добие соодветно време на CPU
  - Спроведување на политиката за распоредување
  - Баланс – сите делови на системот да работат

- За пакетни системи
  - Максимизирај проток
  - Минимизирај време на завршување (од влегување во системот до терминирање)
  - CPU искористеност – CPU да работи цело време

Цели на алгоритамите за распоредување (2)

- Интерактивни системи
  - Време на одзив – одговори на барања брзо
  - Пропорционалност – според корисничките барања

- Системи за работа во реално време
  - Почитувај рокови – без губење податоци
  - Предвидливост – избегни губење квалитет во мултимедијални системи

Распоредување во Batch-системите

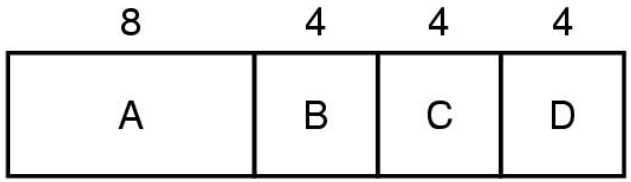
- Прв – Дошол Прв – Услужен (Firs-Come First-Served)
- Најкратката задача прва (Shortest Job First)
- Најкраткото преостанато време следно (Shortest Remaining Time Next)

Прв – Дошол Прв – Услужен (Firs-Come First-Served)

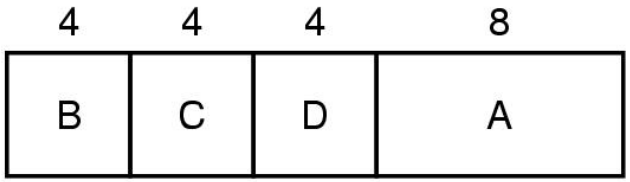
- Наједноставен алгоритам
- Лесно се програмира. Се користи поврзана листа
- Проблеми:
  - Ако има CPU-bound процес со 1 I/O пристап во секунда, и многу I/O-bound процеси кои треба да завршат по 1000 I/O операции, а им треба сосема малку CPU
  - I/O-bound процесите ќе завршат за 1000 сек.
  - Ако се прекинува CPU-bound процесот секои 10ms тогаш I/O-bound процесите ќе завршат за 10 сек.

Најкратката задача прва (Shortest Job First)

- Потреба од познавање на времето на извршување на зададените работи
- Пример за извршување



(a)



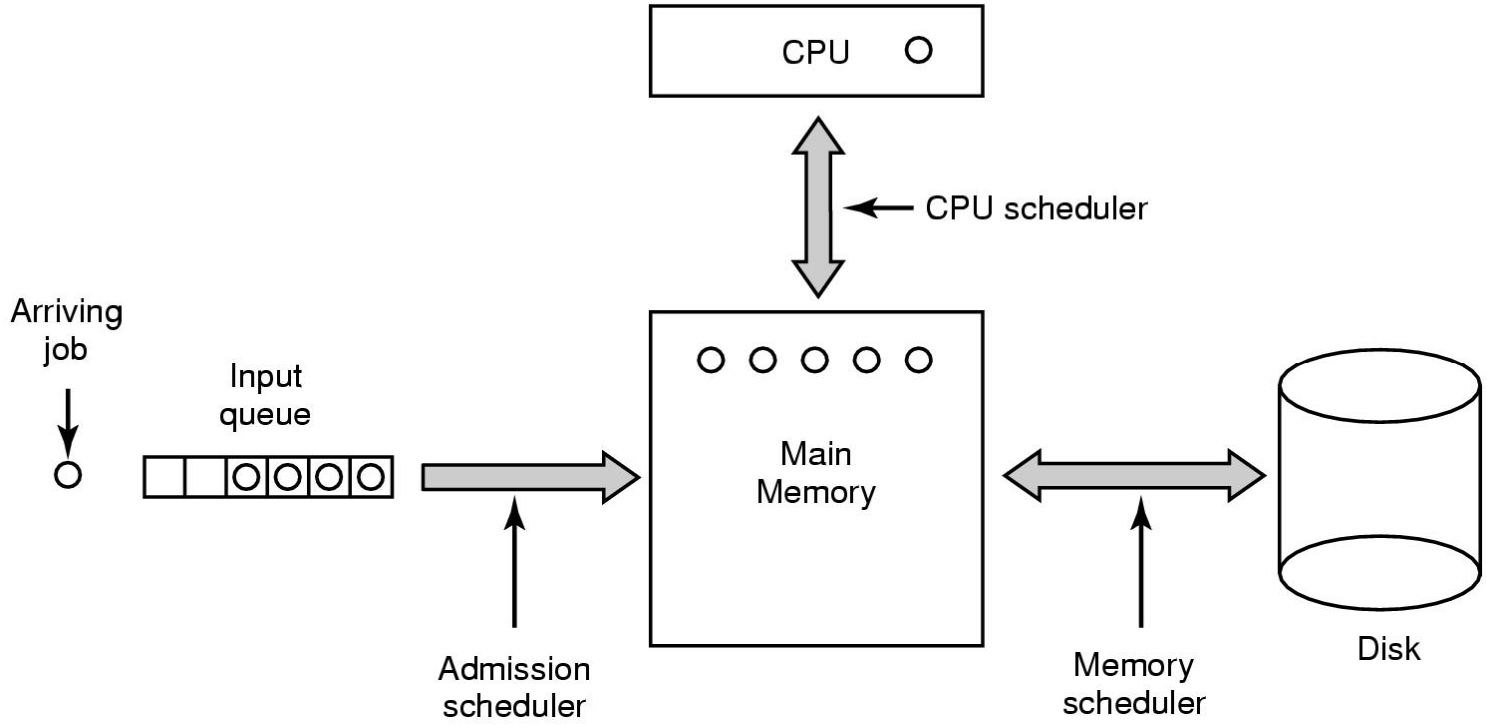
(b)

- Средно време во системот
  - а) (A–8, B –12, C–16, D–20) 14 мин.
  - б) 11 мин.
- Алгоритамот дава најдобри времиња на завршување ако на почетокот се познати сите задачи кои треба да се завршат

Најкраткото преостанато време следно (Shortest Remaining Time Next)

- Треба да се познава времето на извршување
- Се пресметува преостанатото време за извршување

Три нивоа на распоредување во Batch системите



Распоредување во интерактивните системи

- Round-Robin
- Priority Scheduling
- Multiple Queues
- Shortest Process Next
- Guaranteed Scheduling
- Lottery Scheduling
- Fair-Share Scheduling

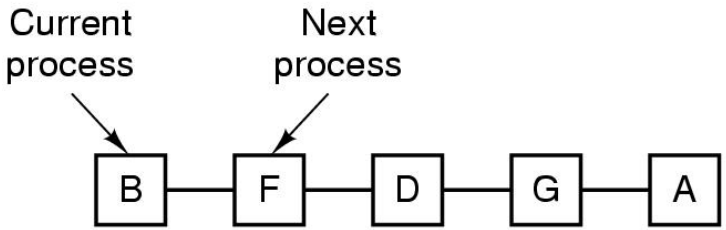
Кружно распоредување – Round Robin (RR)

- За секој процес се одделува мала количина CPU време (*временски квантум)*
  - Откако ќе истече времето процесот се вади од редица на спремни (preempted) и се додава на крајот на редицата на спремни
- Ако има n процеси во редица на спремни и квантумот е q, тогаш секој процес добива $1/n$ од CPU времето во порции од најмногу q временски единици наеднаш. Ниту еден процес не чека повеќе од $(n-1)q$ временски единици.
- Перформанси
  - q големо $\Rightarrow$ FIFO
  - q мало $\Rightarrow$ q мора да е доволно големо во однос на времето за промена на контекст

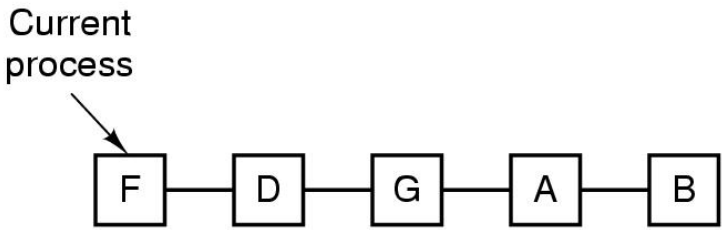
Round-Robin

а) Листа на процеси спремни за извршување

б) Листа на процеси спремни за извршување откако процесот B го искористил своето процесорско време (quantum)

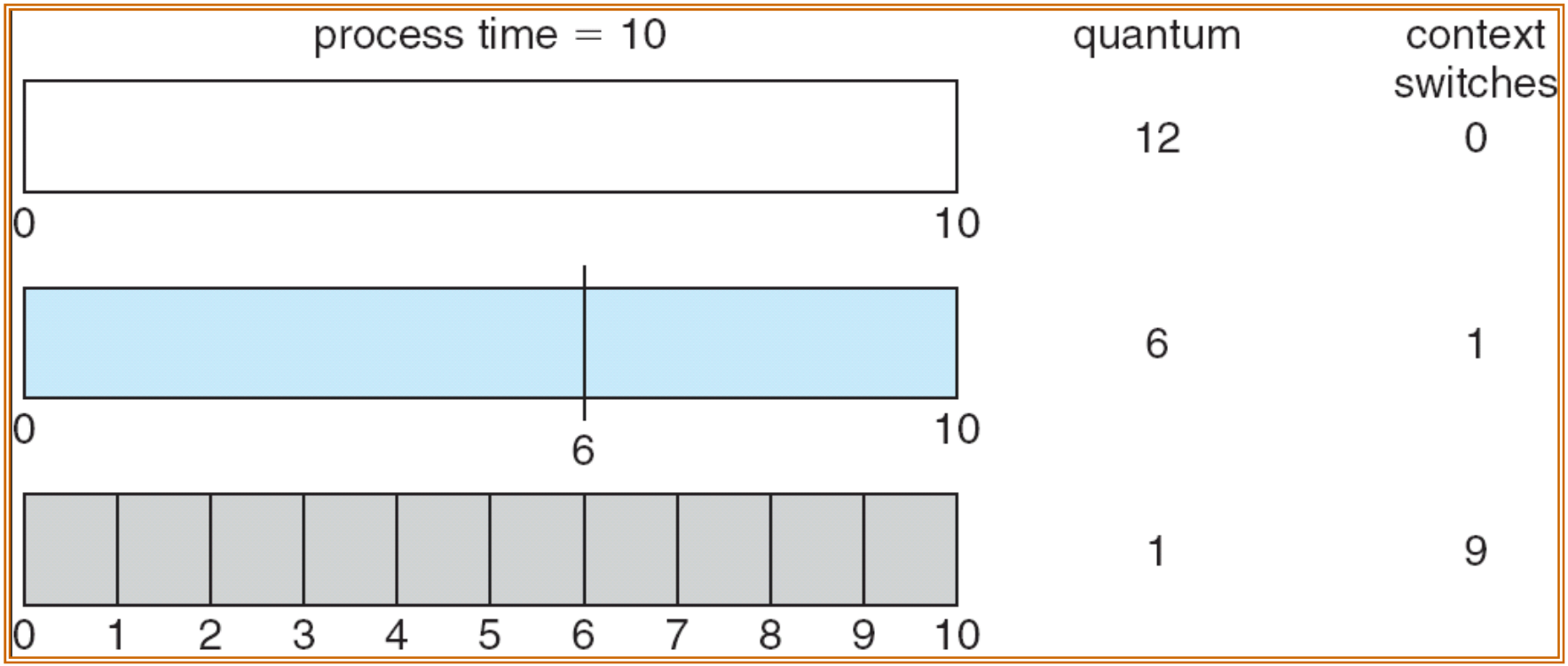


(a)



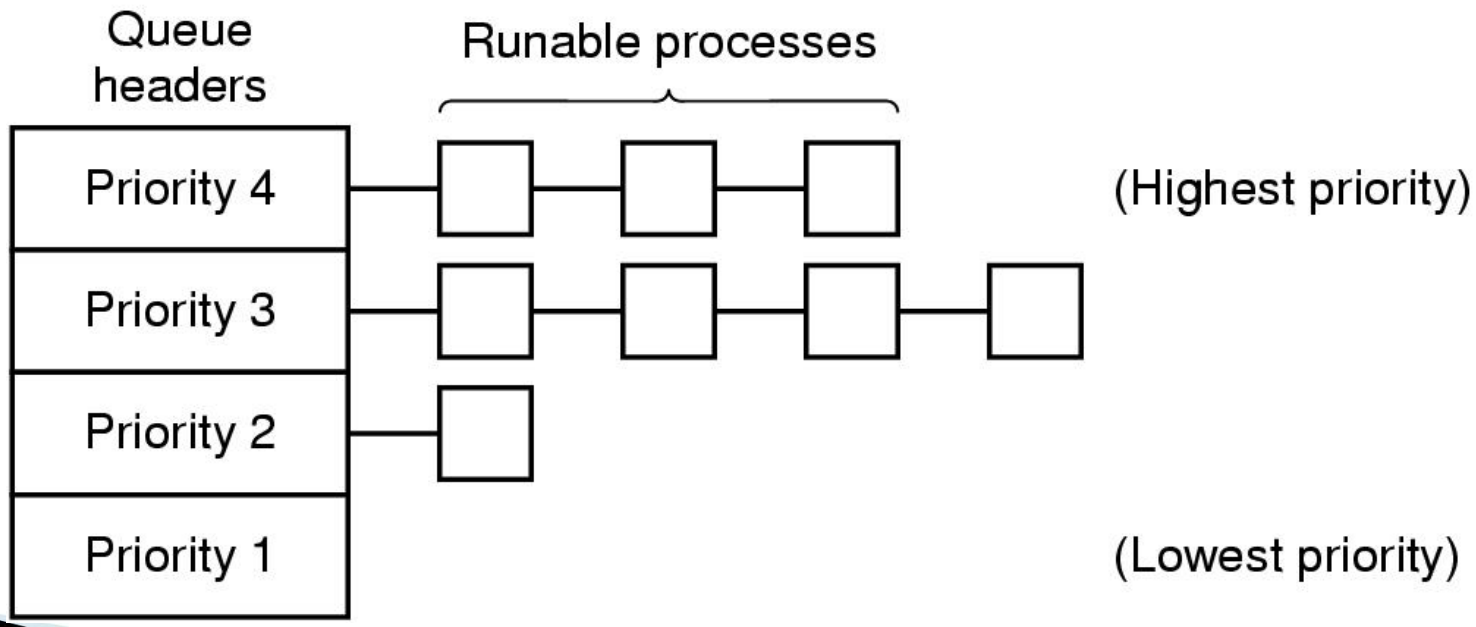
(b)

Временски квантум и време потребно за промена на контекст



Распоредување со приоритети (Priority Scheduling)

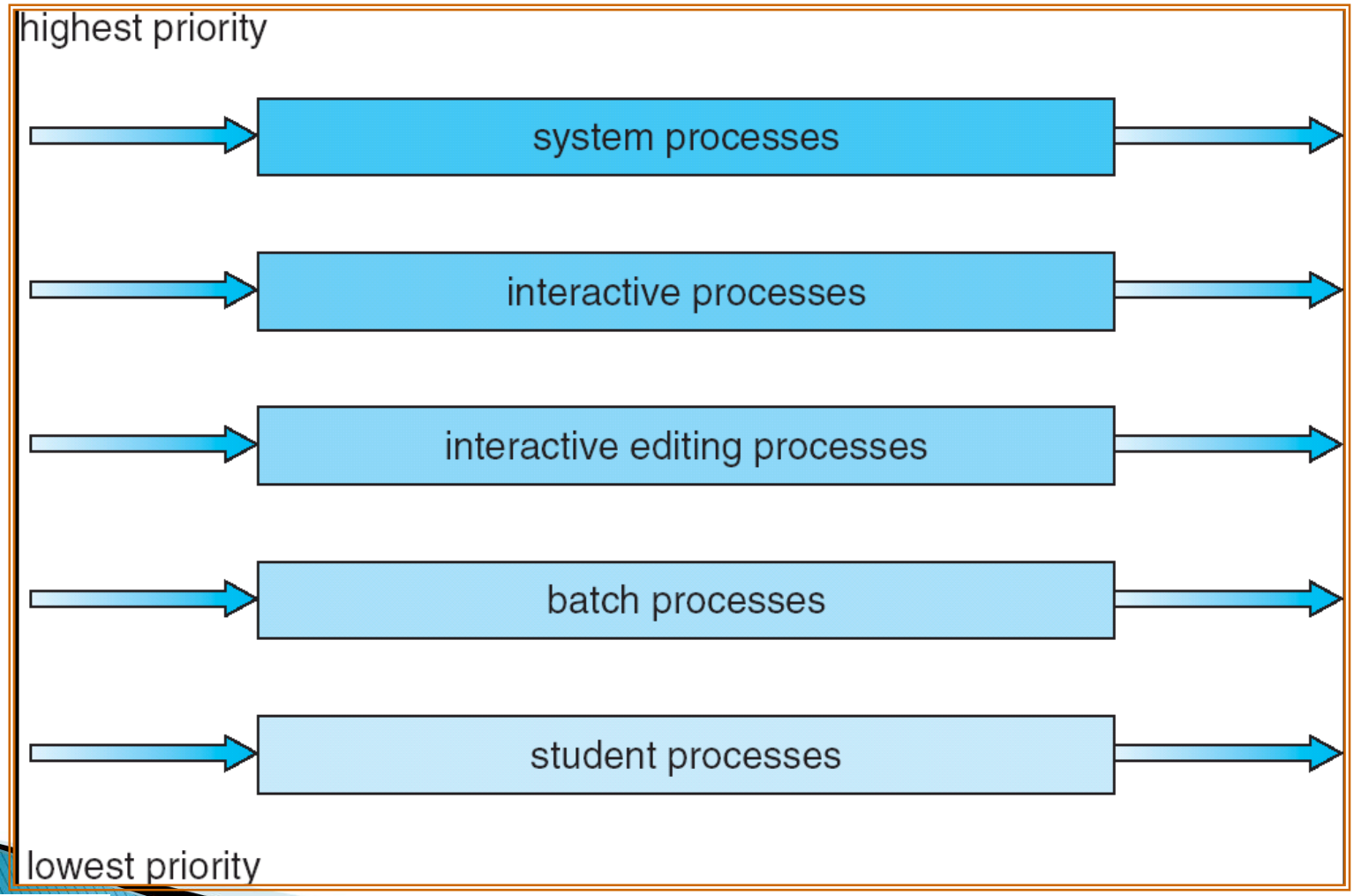
- Дефинирање на различни приоритети на процесите според нивната важност



Распоредување по приоритети

- На секој процес му се придружува цел број кој го означува неговиот приоритет
- CPU се доделува на процесот со најголем приоритет (најмал број ≡ највисок приоритет)
  - SJF е распоредување со приоритети каде приоритетот е одлучен според предвиденото следно време на извршување
- Проблем ≡ Изгладнување – процеси со помал приоритет може никојпат да не се извршат
- Решение ≡ Стареење (Aging) – како поминува времето му се зголемува приоритетот на процесот

Повеќекратни Редови



Приоритет кај повеќекратни Редови

- Намалување на бројот на промени на процеси
- Се дефинираат приоритетни класи и секоја задача добива различно време за извршување

Приоритет	Периоди на извршување
највисок	1
највисок-1	2
највисок-2	4
највисок-3	8

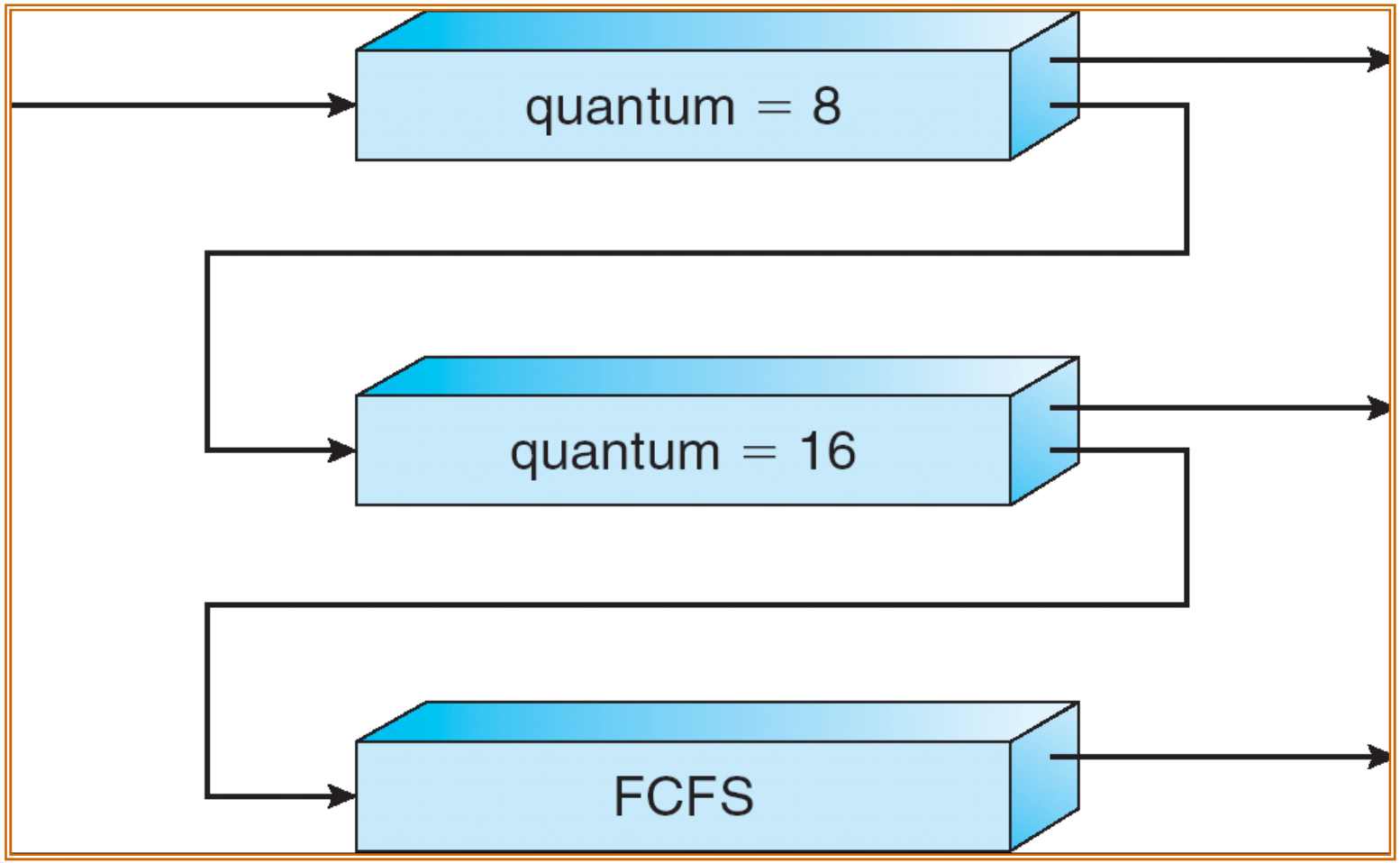
Распоредување кај повеќекратни редови

- Процесот може да има променлив приоритет
- Распоредувачот на повеќекратни редови со повратна информација работи според следните параметри:
  - број на редови
  - распоредувачки алгоритам за секој ред
  - метод кој кажува кога еден процес добива повисок приоритет
  - метод кој кажува кога му се намалува приоритетот на процес
  - метод кој одлучува во кој ред се сместува процесот кога му треба опслужување

Пример

- 3 редови:
  - Q_0 – RR со квантум 8 ms
  - Q_1 – RR со квантум 16 ms
  - Q_2 – FCFS
- Распоредување
  - Нов процес влегува во редот Q_0. Кога ќе добие CPU, задачата добива 8 ms. Ако не заврши за 8 ms, се преместува во Q_1.
  - Во Q_1 задачата се распоредува и добива дополнителни 16 ms. Ако не се заврши, се преместува во Q_2.

Multilevel Feedback Queues



Најкраткиот процес следен (Shortest Process Next)

- Верзија на SJF за интерактивни системи
- Проблем: кој од процесите кои тековно работат е најкус?!?
- Секоја команда се разгледува како посебна работа и се мери нејзиното време на извршување
- Врз основа на проценка прво се извршува најкратката команда

Едноставна проценка на време на извршување (Aging)

- Следното време на опслужување со CPU на еден процес може да се предвиди според времето на опслужување на истиот процес во минатото

- Нека

  T_n е должина на n-то опслужување на тој процес (најсвежа информација, скорешна историја – recent history),

  F_{n+1} е нашата предвидена вредност за следната должина на процесот (F_n е историја од минатото –past history)

  Дефинираме рекурентна врска:

  $$F_{n+1} = \alpha \cdot T_n + (1 - \alpha) \cdot F_n,$$

  (α го контролира влијанието на скората историја од минатото)

Едноставна проценка на време на извршување (Aging) (2)

- Проценка на времетраење на следно извршување на процес :
- Нека имаме две претходни вредности за времето на извршување на еден процес, T_0 и T_1

  $F_1 = T_0$, $F_2 = \alpha \cdot T_1 + (1 - \alpha) \cdot F_1$,

  $F_K = \alpha \cdot T_{K-1} + (1 - \alpha) \cdot F_{K-1}$, $K = 3, 4, \ldots$

  за $\alpha = 1/2$ (скората и историјата од минатото нека имаат еднаква тежина)

  T_0, $T_0/2 + T_1/2$, $T_0/4 + T_1/4 + T_2/2$, $T_0/8 + T_1/8 + T_2/4 + T_3/2$, ...

  (историјата се помалку влијае на иднината)
- – SJF е единствен оптимален (според времето на чекање) кога сите процеси доаѓаат во ист момент, во спротивно не е

Гарантирано распоредување (Guaranteed Scheduling)

- Потреба од гарантирање дека секој корисник ќе добие одредено процесорско време
- Се мери потрошеното време од секој корисник
- Се одредуваат приоритети за процесите врз основа на сообносот на реално потрошеното време и предефинираното време

Распоредување со Лотарија (Lottery Scheduling)

- За едноставна реализација на распоредување во кое ќе се доделат некои предефинирани проценти на процесорско време се користи распоредување со Лотарија
- Секој процес/корисник добива одреден број на тикети кои му овозможуваат користење на даден ресурс
- Кога ќе дојде време за распоредување на нов процес да го користи ресурсот се генерира случаен број во опсегот од 1 до бројот на тикети. Процесот кој го има бараниот број на тикет го добива ресурсот
- Процесите имаат по повеќе тикети. Бројот на тикети е право пропорционален со веројатноста дека тие ќе бидат избрано, односно ќе го добијат ресурсот

Фер-Деливо распоредување (Fair-Share Scheduling)

- Се користи за решавање на проблемот кога даден корисник за да добие повеќе процесорско време може да стартува повеќе процеси
- Идејата е да се извршува по еден процес од секој корисник
- Ако еден корисник има 4 процеси А,Б,В,Г а друг има еден процес Д тогаш процесите ќе се извршуваат по следниов редослед
  - АДБДВДГДАД...
- Ако првиот корисник треба да добие двапати повеќе време тогаш извршувањето би било
  - АБДВГДАБДВГД...

Распоредување во системи кои работат во реално време

- Главна цел е работата да се заврши во дадениот рок
- Мора да се пресмета дали перформансите на компјутерскиот систем задоволуваат
  - Ако има *m* периодични настани
  - Настанот *i* се случува со периода P_i и за да се опслужи бара C_i секунди
- Тогаш оптоварувањето може да е поднесе ако

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО