UDC 004.42
SCOPUS CODE 1710

ოპერატიული მეხსიერების მართვის პროცესების ვიზუალიზების ალგორითმები

ლ. გაჩეჩილაძე კომპიუტერული ინჟინერიის დეპარტამენტი, საქართველოს ტექნიკური უნივერსიტეტი, საქართველო, 0160, თბილისი, მ. კოსტავას 77
E-mail: lia.gachechiladze@mail.ru

რ. სამხარაძე კომპიუტერული ინჟინერიის დეპარტამენტი, საქართველოს ტექნიკური უნივერსიტეტი, საქართველო, 0160, თბილისი, მ. კოსტავას 77
E-mail: r.samkharadze@ mail.ru

მ. ქურდაძე ტელეკომუნიკაციის დეპარტამენტი, საქართველოს ტექნიკური უნივერსიტეტი, საქართველო, 0160, თბილისი, მ. კოსტავას 75
E-mail: kurdadze14@mail.ru

რეცენზენტები:

მ. კიკნაძე, სტუ-ის ინფორმატიკისა და მართვის სისტემების ფაკულტეტის პროფესორი
E-mail: mziakiknadze@mail.ru

ა. ბენაშვილი, სტუ-ის ინფორმატიკისა და მართვის სისტემების ფაკულტეტის პროფესორი
E-mail: sbenashvili@yahoo.com

ანოტაცია. ოპერატიული მეხსიერების მართვის პროცესების ვიზუალიზების ალგორითმები ახდენენ ოპერატიული მეხსირების განაწილების სამივე სტრატეგიის რეალიზებას: "პირველი შესაფერისი", "ყველაზე შესაფერისი" და "ნაკლებად შესაფერისი". თითოეული სტრატეგიის რელიზებისთვის ალგორითმები შემუშავებულია როგორც თანაბარი პრიორიტეტის, ისე არათანაბარი პრიორიტეტის მქონე პროცესებისთვის. შემუშავებული ალგორითმები შესაძლებელს ხდის ოპერატიული მეხსიერების მეზობელი და არამეზობელი უბნების გაერთიანების პროცესის, მეხსიერების საწყისი და საბოლოო უბნების მისამართების ცვლილების პროცესის, აგრეთვე პროგრამების მიერ დაკავებული ოპერატიული მეხსიერების უბნების ზომების ცვლილების პროცესების მართვას. მოყვანილი ალგორითმების საფუძველზე აგებულია შესაბამისი პროგრამული საწვრთნელი, რომელიც იძლევა ოპერატიული მეხსიერების განაწილების სამივე სტრატეგიის რეალიზების შესაძლებლობას.

საკვანძო სიტყვები: ოპერატიული მეხსიერება, ვიზუალიზება, პროცესი, ალგორითმი.

შესავალი

ოპერაციული სისტემების შესწავლისას ერთ-ერთი პრობლემაა ის, რომ მათი მუშაობის დროს მიმდინარე პროცესები ადამიანის თვალისთვის უხილავია [1-5]. გარდა ამისა, ამ პროცესებს შემთხვევითი ხასიათი აქვს. ამიტომ, აქტუალურია ოპერაციული სისტემის მუშაობის დროს კომპიუტერში მიმდინარე პროცესების ვიზუალიზება. აქედან გამომდინარე, საჭიროა სათანადო მიდგომებისა და მოდელების შემუშავება და სასწავლო პროცესის დახვეწა.

ძირითადი ნაწილი

იმისათვის, რომ მოვახდინოთ ოპერატიული მეხსიერების მართვის მოდელირება უნდა გავაკეთოთ შემდეგი დაშვებები [6-10]:

1. ვიცით სისტემაში თითოეული პროცესის შემოსვლის დრო.
2. ვიცით თითოეული პროცესი როდის და რა ზომის მეხსიერებას მოითხოვს.
3. ვიცით ოპერაციული სისტემის მიერ პროცესისათვის გამოყოფილი მეხსიერების ზომა.

ამ დაშვებების გათვალისწინებით მომლოდინე პროცესები შეგვიძლია წარმოვადგინოთ შემდეგი მიმდევრობის სახით:

$P_1(t_{ლ1}, t_{ა1}, z_1)$	$P_2(t_{ლ2}, t_{ა2}, z_2)$	$P_3(t_{ლ3}, t_{ა3}, z_3)$	...	$P_i(t_{ლi}, t_{აi}, z_i)$	*

პროცესი წარმოვადგინოთ ობიექტის სახით. მას აქვს შემდეგი მახასიათებლები: ლოდინის დრო, აქტიურობის დრო, მოთხოვნილი მეხსიერების ზომა და პრიორიტეტი. როგორც ვხედავთ, მიმდევრობის პირველი ელემენტი შეიცავს ინფორმაციას პირველი პროცესის შესახებ, მეორე ელემენტი – მეორე პროცესის შესახებ და ა.შ. მიმდევრობის პირველ ელემენტში მოთავსებულია $P_i(t_{ლi}, t_{აi}, z_i)$. აქ P_i არის i-ური პროცესის იდენტიფიკატორი. $t_{ლi}$ არის P_i პროცესის ლოდინის დრო. $t_{აi}$ არის ის დრო, რომლის განმავლობაშიც P_i პროცესი იკავებს ოპერატიული მეხსიერების z_i ზომის უბანს. განვიხილოთ მაგალითი. დავუშვათ გვაქვს პროცესების შემდეგი მიმდევრობა:

$P_1(0,3,100)$	$P_2(0,1,250)$	$P_3(2,5,230)$	$P_4(1,2,110)$	$P_5(0,1,180)$	*

მიმდევრობის პირველ ელემენტში მოთავსებულია "$P_1(0,3,100)$". ეს იმას ნიშნავს, რომ P_1 პროცესი დროის 3 ერთეულის განმავლობაში ითხოვს მეხსიერების 100 ერთეულს და იღებს მას თუ არსებობს მეხსიერების შესაბამისი ზომის უბანი. 100 ერთეული შეიძლება იყოს 100 კილობაიტი ან 100 მეგაბაიტი და ა. შ. P_1 პროცესი მზა პროცესების რიგში დაუყოვნებლივ დგება, რადგან მისი ლოდინის დრო 0-ის ტოლია. სისტემაში დაუყოვნებლივ შემოდის, აგრეთვე, P_2 პროცესი, რომელიც დროის 1 ერთეულის განმავლობაში ითხოვს მეხსიერების 250 ერთეულს. P_2 პროცესი იღებს მას თუ არსებობს ამ ზომის უბანი. წინააღმდეგ შემთხვევაში, პროცესი ელოდება მეხსიერების საჭირო ზომის უბნის გათა-

ვისუფლებას. დროის 2 ერთეულის გავლის შემდეგ სისტემაში P_3 პროცესი შემოდის. დროის კიდევ 1 ერთეულის გავლის შემდეგ სისტემაში P_4 პროცესი შემოდის. P_4 პროცესთან ერთად სისტემაში შემოდის, აგრეთვე, P_5 პროცესი, რადგან მისი ლოდინის დრო 0-ის ტოლია. როგორც კი ამოიწურება P_1 პროცესისათვის გამოყოფილი დროის 3 ერთეული, ის ათავისუფლებს მის მიერ დაკავებული მეხსიერების 100 ერთეულს და გადის სისტემიდან.

პროცესის შესრულების ასეთი სახით წარმოდგენა შეგვიძლია განვიხილოთ როგორც პროცესის შესრულებისა და მისთვის მეხსიერების საჭირო ზომის უბნის გამოყოფის გამარტივებული იმიტაციური მოდელი. იგი საშუალებას გვაძლევს შევიმუშაოთ პროცესისათვის მეხსიერების საჭირო ზომის უბნის გამოყოფის ვიზუალიზების ალგორითმები.

მოყვანილი მოდელის საფუძველზე შემუშავებულია პროცესებს შორის რესურსების განაწილების ალგორითმები იმ შემთხვევებისათვის, როცა პროცესებს აქვთ თანაბარი და არათანაბარი პრიორიტეტები. ორივე შემთხვევისათვის შემუშავებულია ალგორითმების სიმრავლე:

$$L = \{ L_1, L_2, L_3, L_4, L_5, L_6, L_7 \}.$$

აქ L_1 არის პროცესის შესრულების ალგორითმი მომლოდინე პროცესების რიგში შემოსვლის მომენტიდან შესრულების დამთავრებამდე, L_2 არის პროცესისთვის მეხსიერების გამოყოფის ალგორითმი “პირველი შესაფერისი” დისციპლინის მიხედვით, L_3 არის პროცესისთვის მეხსიერების გამოყოფის ალგორითმი “ყველაზე შესაფერისი” დისციპლინის მიხედვით, L_4 არის პროცესისთვის მეხსიერების გამოყოფის ალგორითმი “ნაკლებად შესაფერისი” დისციპლინის მიხედვით, L_5 არის პროცესის მიერ მეხსიერების გათავისუფლების ალგორითმი, L_6 არის მეხსიერების ორი თავისუფალი მეზობელი უბნის გაერთიანების ალგორითმი, L_7 არის მეხსიერების არამეზობელი თავისუფალი უბნების გაერთიანების ალგორითმი.

დავუშვათ, პროცესებს თანაბარი პრიორიტეტები აქვს: $П_1 > П_2 > \ldots > П_l$.

შესაბამისად, მზა პროცესების სიაში შემოსული რიგში დგება FIFO დისციპლინის მიხედვით. ჯერ განვიხილოთ პროცესისთვის “პირველი შესაფერისი” დისციპლინის მიხედვით მეხსიერების გამოყოფის L_2 ალგორითმი. ის შემდეგი ბიჯებისაგან შედგება:

1. შესრულდება მეხსიერების თავისუფალი უბნების დახარისხება F_b საწყისი მისამართების ზრდადობის მიხედვით.

2. P_i პროცესის ზომა z_i შედარდება თავისუფალი უბნების ზომების სიმრავლის – $\{ F_1, F_2, F_3, \ldots, F_j \}$ თითოეულ ელემენტს.

3. სიმრავლიდან გამოეყოფა პირველივე მისაღები – F_m, რომლის ზომა P_i პროცესის ზომაზე მეტია ან ტოლი, $F_1 \geq z_i$.

4. P_i პროცესს გამოეყოფა F_1 ზომის უბანი, რომლის საწყისი მისამართია $A_{bi} = F_{b1}$, ბოლო მისამართი კი - $A_{ei} = A_{bi} + z_i - 1$.

5. შეიცვლება დარჩენილი თავისუფალი მეხსიერების ზომა – $F_k = F_k - z_i$. მისი საწყისი მისამართი იქნება $F_{b1} = A_{ei} + 1$, ბოლო მისამართი F_{e1} – კი იგივე დარჩება.

განვიხილოთ პროცესისთვის "ყველაზე შესაფერისი" დისციპლინის მიხედვით მეხსიერების გამოყოფის L_3 ალგორითმი. მისი ბიჯებია:

1. სრულდება მეხსიერების თავისუფალი უბნების დახარისხება F_b საწყისი მისამართების ზრდადობის მიხედვით.

2. გამოითვლება სხვაობები თითოეული თავისუფალი უბნის ზომასა და მოცემული პროცესის ზომას შორის, $D_k = F_k - z_i$. მიიღება სხვაობების სიმრავლე: { $D_1, D_2, D_3, \dots, D_j$ }.

3. P_i პროცესს გამოეყოფა მინიმალური ზომის უბანი: $F^{min}_k = \min\{D_1,D_2,D_3,\dots,D_j\}$.

4. P_i პროცესს გამოეყოფა F_k ზომის უბანი, რომლის საწყისი მისამართია$A_{bi} = F_{bk}$, ბოლო მისამართი კი – $A_{\partial i} = A_{bi} + z_i - 1$.

5. შეიცვლება დარჩენილი თავისუფალი მეხსიერების ზომა – $F_k = F_k - z_i$. მისი საწყისი მისამართი იქნება $F_{bk} = A_{\partial i} + 1$, ბოლო მისამართი $F_{\partial k}$ – k_i იგივე დარჩება.

განვიხილოთ პროცესისთვის "ნაკლებად შესაფერისი" დისციპლინის მიხედვით მეხსიერების გამოყოფის L_4 ალგორითმი. მისი ბიჯებია:

1. სრულდება მეხსიერების თავისუფალი უბნების დახარისხება F_s საწყისი მისამართების ზრდადობის მიხედვით.

2. გამოითვლება სხვაობები თითოეული თავისუფალი უბნის ზომასა და მოცემული პროცესის ზომას შორის, $D_k = F_k - z_i$. მიიღება სხვაობების სიმრავლე { $D_1, D_2, D_3, \dots, D_j$ }.

3. P_i პროცესს გამოეყოფა მაქსიმალური ზომის უბანი:

$F^{max}_k = \max\{ D_1, D_2, D_3, \dots, D_j \}$.

4. P_i პროცესს გამოეყოფა F_k ზომის უბანი, რომლის საწყისი მისამართია $A_{bi} = F_{bk}$, ბოლო მისამართი კი – $A_{\partial i} = A_{bi} + z_i - 1$.

5. შეიცვლება დარჩენილი თავისუფალი მეხსიერების ზომა – $F_k = F_k - z_i$. მისი საწყისი მისამართი იქნება $F_{bk} = A_{\partial i} + 1$, ბოლო მისამართი $F_{\partial k}$ – კი იგივე დარჩება.

განვიხილოთ პროცესის მიერ მეხსიერების გათავისუფლების L_5 ალგორითმი. ის შემდეგი ბიჯებისაგან შედგება:

1. თავისუფალი უბნების ზომების F სიმრავლეს დაემატება ახალი უბანი, რომლის ზომაა P_i პროცესის ზომა z_i და საწყისი და ბოლო მისამართებია A_{bi} და $A_{\partial i}$, $F_{bk} = A_{bi}$ და $F_{\partial k} = A_{\partial i}$, $F_k = z_i$.

2. მოწმდება, მეხსიერების უბნის გათავისუფლების შედეგად, ორი მეზობელი უბანი განლაგდა ერთმანეთის მეზობლად თუ არა.

3. თუ მეხსიერების ორი თავისუფალი უბანი ერთმანეთის მეზობლად აღმოჩნდა, მაშინ მუშაობას იწყებს L_6 ალგორითმი.

განვიხილოთ ორი თავისუფალი მეზობელი უბნის გაერთიანების L_6 ალგორითმი. ის შემდეგი ბიჯებისაგან შედგება:

1. პირველი თავისუფალი უბნის ბოლო მისამართი შედარდება მეორე თავისუფალი უბნის საწყის მისამართს. თუ უკანასკნელი მისამართი პირველზე 1-ით მეტია, მაშინ ეს ორი თავისუფალი უბანი ერთმანეთის მეზობლად მდებარეობს – $F_{ks} = F_{k-1b} + 1$.

2. სრულდება მათი გაერთიანება ანუ მისამართების გადაწყობა. კერძოდ, პირველი მეზობელი უბნის საწყისი მისამართი იგივე რჩება, ხოლო მეორე უბნის ბოლო მისამართი გახდება პირველი უბნის ბოლო მისამართი, $F_{k-1b} = F_{k\partial}$.

„განვიხილოთ მეხსიერების არამეზობელი თავისუფალი უბნების გაერთიანების L_7 ალგორითმი. ის შემდეგი ბიჯებისაგან შედგება:

1. მეხსიერების პირველი თავისუფალი უბნის საწყისი მისამართი შედარდება თითოეული პროცესის მიერ დაკავებული მეხსიერების უბნის საწყის მისამართებს. ამოირჩევა ის უბანი, რომლის

მისამართი ყველაზე ახლოსაა პირველი თავისუფალი უბნის საწყისი მისამართთან. ამასთან, თავისუფალი უბნის საწყისი მისამართი უნდა იყოს ნაკლები პროცესის მიერ დაკავებული უბნის საწყის მისამართზე, $A_{ბi} > A_{ბk}$.

2. მეხსიერების ეს ორი უბანი ადგილებს ცვლის, ეს ხორციელდება მათი საწყისი და ბოლო მისამართების ცვლილებით:

$A_{ბi} = F_{ბk}$, $A_{მi} = A_{bi} + z_i - 1$, $F_{ბk} = A_{მi} + 1$, $F_{მk} = F_{ბk} + F_k - 1$.

3. ეს პროცესი მეორდება მანამ, სანამ ერთმანეთის მეზობლად არ აღმოჩნდება ორი თავისუფალი უბანი. ამ შემთხვევაში მუშაობას იწყებს ორი მეზობელი თავისუფალი უბნის გაერთიანების L_6 ალგორითმი.

4. აღნიშნული სამი ეტაპი მეორდება მანამ, სანამ თავისუფალი არ დარჩება მეხსიერების ერთი უბანი, რომელიც მეხსიერების ზედა მისამართებს იკავებს.

განვიხილოთ L_1 ალგორითმი. ის შემდეგი ბიჯებისაგან შედგება:

1. მომლოდინე პროცესების რიგიდან შესრულდება P_i პროცესი შესაბამისი ელემენტების წაკითხვა. თუ მისი ლოდინის დრო ნულის ტოლია, $t_{ლi} = 0$, მაშინ ის დგება მზა პროცესების სიაში. თუ $t_{ლi} \neq 0$, მაშინ ის რჩება მომლოდინე პროცესების რიგში ლოდინის დროის განულებამდე.

2. მოწმდება მიმდევრობის უკანასკნელი ელემენტი არის თუ არა "*". თუ არის, ეს იმას ნიშნავს, რომ მომლოდინე პროცესები აღარ გვაქვს. წინააღმდეგ შემთხვევაში, გადავდივართ მე-3 ბიჯზე.

3. P_i პროცესი დგება მზა პროცესების სიის ბოლოში.

4. ზემოთ მოყვანილი ყველა ნაბიჯი მეორდება თითოეული პროცესისათვის. შედეგად, შეივსება მზა პროცესების სია.

5. თუ P_i პროცესი პირველია მომლოდინე პროცესების სიაში და ოპერაციულ სისტემას აქვს საკმარისი ზომის მეხსიერება მისთვის გამოსაყოფად ანუ $F_k \geq z_i$, მაშინ მას გამოეყოფა z_i ზომის უბანი მეხსიერებაში.

6. ამ მომენტში მუშაობას იწყებს პროცესისთვის მეხსიერების გამოყოფის L_2, L_3 ან L_4 ალგორითმი.

7. მე-5 და მე-6 ნაბიჯი მეორდება თითოეული პროცესისათვის. შედეგად, შეივსება პროცესებისთვის გამოყოფილი მეხსიერების უბანი.

8. ამის შემდეგ, იწყება დროის ათვლა. თითოეული პროცესის აქტიურობის დროს t_{ai} ერთდროულად აკლდება დროის ერთი ერთეული მანამ, სანამ რომელიმე მათგანის აქტიურობის დრო არ განულდება, $t_{ai} = 0$.

9. თუ $t_{ai} = 0$, მაშინ P_i ათავისუფლებს მეხსიერების დაკავებულ უბანს და გადის სისტემიდან.

10. მუშაობას იწყებს პროცესის მიერ მეხსიერების გათავისუფლების L_5 ალგორითმი.

11. მოწმდება მეხსიერებაში არის თუ არა თავისუფალი მეზობელი უბნები. თუ არის მაშინ მუშაობას იწყებს ორი თავისუფალი მეზობელი უბნების გაერთიანების L_6 ალგორითმი.

12. მოწმდება მზა პროცესების სია. თუ მასში არის პროცესი, მაშინ მოწმდება მეხსიერებაში არის თუ არა თავისუფალი ზომის უბანი მის მოსათავსებლად. თუ არის, მაშინ პროცესი მოთავსთდება მეხსიერებაში. თუ არ არის, მაშინ სრულდება სრულდება მეხსიერების თავისუფალი არამეზობელი უბნების გაერთიანება ანუ მუშაობას იწყებს L_7 ალგორითმი. თუ მაინც არ აღმოჩნდა თავისუფალი მეხსიერების საკმარისი ზომის უბანი, მაშინ პროცესი რჩება მზა პროცესების სიაში.

13. თუ მეხსიერებაში არის ერთი პროცესი მაინც, მაშინ გადავდივართ მე-8 ბიჯზე. წინააღმდეგ შემთხვევაში ალგორითმი მუშაობას ამთავრებს.

რაც შეეხება მეხსიერების მართვის ალგორითმს არათანაბარი პრიორიტეტის მქონე პროცესებისათვის, ამ შემთხვევაში $П_1 > П_2 > \ldots > П_l$. შესაბამისად, მზა პროცესების სიაში შემოსული პროცესები რიგში პრიორიტეტის მიხედვით დგება. ზემოთ მოყვანილი ალგორითმები ძალაშია არათანაბარი პრიორიტეტების შემთხვევაშიც.

შემოთავაზებული ვიზუალიზების მოდელისა და წარმოდგენილი ალგორითმების საფუძველზე რეალიზებულია შესაბამისი პროგრამული საწვრთნელი და შესაბამისი გრაფიკული ინტერფეისი.

დასკვნა

ამრიგად, სტატიაში შემოთავაზებულია ოპერატიული მეხსიერების მართვის პროცესების ვიზუალიზების ალგორითმები, რომელთა გამოყენებით შესაძლებელია ოპერატიული მეხსირების განაწილების სტრატეგიების მოდელირება: “პირველი შესაფერისი”, “ყველაზე შესაფერისი” და “ნაკლებად შესაფერისი”. თითოეული მათგანისთვის ალგორითმები შემუშავებულია როგორც თანაბარი პრიორიტეტის, ისე არათანაბარი პრიორიტეტის მქონე პროცესებისთვის. მოყვანილი ალგორითმების საფუძველზე აგებულია შესაბამისი პროგრამული საწვრთნელი, რომელიც იძლევა ოპერატიული მეხსიერების განაწილების პროცესების ეფექტურად სწავლების შესაძლებლობას.

ლიტერატურა

1. Deitel H. Introduction to operating systems: 2 Vols. T. 1. M.: “Mir”. 1987, 359 p. (in Russian).
2. Deitel H. Introduction to Operating Systems: 2 Vols. T. 2. M.: “Mir”. 1987, 398 p. (in Russian).
3. Tanenbaum A. Modern operation systems. St. Petersburg: “Piter”. 2002, 1040 p. (in Russian).
4. Samkharadze R., Kevkhishvili M., Gachechiladze L. On the issue of building an intellectual environment for distance learning and testing. XIV international symposium “Management of Large Systems”. Proceedings of the symposium “Informatization, automation, management”. 2000. (in Russian).
5. Samkharadze R. The Petry nets in computer learning. The monograph. Tbilisi. “Technical University”. ISBN 978-99940-57-93-1. 2007, 156 p. (in Georgian).
6. Samkharadze R, Gachechiladze L. Imitation models in computer learning. Tbilisi. “Technical University”. ISBN 978-9941-14-057-0. 2008, 124 p. (in Georgian).
7. Samkharadze R., Gvaramia E., Gachechiladze L. Modeling of the process statuses with Petri networks. Proceedings. Automated control systems. # 1(4). Tbilisi. “Technical University”. ISSN 1512-3979. 2008. 84-88 pp. (in Georgian).
8. Samkharadze R., Gvaramia E., Gachechiladze L. Methods of implementing the active learning method for teaching of changes the process states. Collection of reports of the international scientific conference "Information technologies 2008". Tbilisi. ISBN 978-994114-191-1. 298-301 pp. (in Russian).

9. Samkharadze R, Gachechiladze L, Kurdadze M. Queue theory in computer learning. The monograph. Tbilisi. GTU "IT-consulting scientific center". ISBN 978-9941-0-9643-3. 2017, 93 p. (in Georgian).
10. Samkharadze R, Kevkhishvili, Kamkamidze E. Modern technologies on physics lessons. GTU internatioonal-scientific conference "Verbal communication technologies - 4". Tbilisi. 2014. (in Georgian).

UDC 004.42
SCOPUS CODE 1710

Drawing algorithms for RAM memory management

L. Gachechiladze Department of Computer Engineering, Georgian Technical University, 77 M. Kostava str, 0160 Tbilisi, Georgia
E-mail: lia.gachechiladze@mail.ru

R. Samkharadze Department of Computer Engineering, Georgian Technical University, 77 M. Kostava str, 0160 Tbilisi, Georgia
E-mail: r.samkharadze@ mail.ru

M. Kurdadze Department of Telecommunication, Georgian Technical University, 75 M. Kostava str, 0160 Tbilisi, Georgia
E-mail: kurdadze14@mail.ru

Reviewers:

S. Benashvili, Professor, Faculty of Informatics and Control Systems, GTU
E-mail: mziakiknadze@mail.ru
S. Benashvili, Professor, Faculty of Informatics and Control Systems, GTU
E-mail: sbenashvili@yahoo.com

Abstract. The article offers new approach to questions of visualization of RAM memory management. It is offered corresponding model which is based on three assumptions: 1. It is known the receipt time in system of each process; 2. The size of memory requested by the process and inquiry time is known; 3. The size of memory allocated to process by operative system is known. The corresponding algorithms of the visualization principles of RAM memory management for the processes with equal and unequal priorities are offered. In particular, the set of algorithms are developed for the realization of principles of allocation of memory: "the first suitable", "the most suitable" and "the less suitable".

Key words: Agorithm; process; RAM memory; visualization.