1. ოპერაციული სისტემების არქიტექტურები

მონოლიტური. მთლიანი სისტემა მუშაობს როგორც ერთი პროგრამა, ანუ ოპერაციული სისტემის კოდი დაწერილია როგორც პროცედურათა ნაკრები, რომელიც ერთ დიდ შესრულებად პროგრამაში არის გაერთიანებული. ამ ტექნოლოგიით ნებისმიერ პროცედურას შეუძლია გამოიძახოს სხვა პროცედურა, რომელმაც შესაძლებელია მისთვის შეასრულოს სასარგებლო სამუშაო. ვინაიდან პროცედურებს გააჩნიათ განუსაზღვრელი წვდომა ერთიმეორეზე და აპარატურაზე სისტემა მთლიანობაში შეიძლება აღმოჩნდეს არამდგრადი შეცდომის ან ზიანის შემცველი კოდის მიმართ.

მიკრობირთვული. მიკრობირთვული არქიტექტურის მიხედვით ოპერაციული სისტემა იყოფა ცალკეულ მოდულებად. ამ მოდულებიდან მხოლოდ ერთი - მიკრობირთვი, მუშაობს ბირთვის რეჟიმში. ყველა მოდული ერთმანეთთან ურთიერთქმედებს მიკრობირთვის მეშვეობით, რომელიც უზრუნველყოფს პროგრამებს შორის კავშირს, პროცესორის გამოყენების დაგეგმვას, წყვეტის პირველად დამუშვებას, შეტანა/გამოტანის ოპერაციებს და მეხსიერების საბაზო მართვას.

ჰიბრიდული. ბირთვის კომპილაციისას შესაძლებელია ბირთვის მრავალი კომპონენტის ე.წ. მოდულების დინამიური ჩატვირთვა/ამოტვირთვა. მოდულის ჩატვირთვის მომენტში მისი კოდი იტვირთება სისტემის დონეზე და უკავშირდება ბირთვის დანარჩენ ნაწილს. მოდულის შიგნით შესაძლებელია იყოს გამოყენებული ბირთვის მიერ ექსპორტირებული ფუნქცია.

2. პროცესები

დემონი. ფონურ რეჟიმში მომუშავე პროცესებს, რომლებიც წარმოქმნილია გარკვეული აქტიური საქმიანობის წარმოებისათვის დემონები ეწოდებათ.

ზომბი. „დასრულდა“ მდგომარეობაში მყოფ პროცესებს UNIX მსგავს ოპერაციულ სისტემაში ეწოდებათ ზომბი-პროცესები (zombie).

კონტექსტის გადართვა. პროცესორის ერთი პროცესიდან მეორეზე კორექტულად გადასართველად აუცილებელია შესრულებადი პროცესის კონტექსტის შენახვა და იმ პროცესის კონტექსტის აღდგენა, რომელზეც იქნება პროცესორი გადართული. პროცესის ქმედუნარიანობის შენახვა/აღდგენის ასეთ პროცედურას კონტექსტის გადართვა ეწოდება.

პროცესის იდენტიფიკატორი. UNIX მსგავს სისტემაში წარმოქმნილი ყოველი პროცესი უნიკალურ სახელად ღებულობს რიცხვით მნიშვნელობას - საიდენტიფიკაციო ნომერს PID (process identifiator). სხვადასხვა სისტემებში პროცესის იდენტიფიცირებისათვის გამოყოფილი რიცხვითი მნიშვნელობები მერყეობს 0-დან გარკვეულ მაქსიმალურ მნიშვნელობამდე.

სისტემაში პროცესისათვის საიდენტიფიკაციო ნომრის მინიჭებისას უნიკალობის შენარჩუნების მიზნით ყოველ ახალ პროცესს სისტემა რიცხვით მნიშვნელობას ანიჭებს ზრდადი მიმდევრობით, ანუ ბოლოს წარმოქმნილი პროცესის საიდენტიფიკაციო ნომერს + 1. პროცესის მიერ დაკავებული

ნომერი მისი დასრულების შემდეგ თავისუფლდება და მისი გამოყენება შესაძლებელია ახალი პროცესისათვის. სისტემაში თუ მიღწეულია საიდენტიფიკაციო ნომრის მაქსიმალური მნიშვნელობა, მაშინ ახალი პროცესს საიდენტიფიკაციო ნომრად ენიჭება პირველივე მინიმალური თავისუფალი რიცხვითი მნიშვნელობა.

მდგომარეობათა დიაგრამა. ამბობენ, რომ პროცესი სრულდება, ანუ იმყოფება მდგომარეობაში „შესრულება“, თუ მისთვის გამოყოფილია ცენტრალური პროცესორი. პროცესი იმყოფება მდგომარეობაში „მზადყოფნა“, თუ მისთვის ცენტრალური პროცესორის გამოყოფის შემთხვევაში პროცესს შეუძლია გააგრძელოს შესრულება. პროცესი იმყოფება მდგომარეობაში „ბლოკირება“, თუ მას არ შეუძლია შესრულების გაგრძელება გარკვეული მოვლენის დადგომამდე.

3. დაგეგმვის ალგორითმები

FCFS. ამ ალგორითმის გამოყენებისას სისტემაში იქმნება მდგომარეობაში მზაყოფნა მყოფი პროცესების მიმდევრობა. სისტემაში წარმოქმნილი ყოველი ახალი პროცესი ამ მიმდევრობაში ემატება ბოლოდან. შესრულებას იწყებს მიმდევრობის თავში მყოფი პირველივე პროცესი. პროცესს პროცესორი გამოეყოფა იმ დროითი შუალედით რამდენიც საჭიროა მის შესასრულებლად. პროცესი პროცესორს გამოანთავისუფლებს მხოლოდ იმ შემთხვევაში, თუ მან დასრულდა ან გარკვეული მიზეზით მოხდა მისი ბლოკირება. მას შემდეგ რაც სისტემაში დაფიქსირდება მოვლენა, რომელსაც ბლოკირებული პროცესი ელოდებოდა ის გადადის მდგომარეობაში მზადყოფნა და მიმდევრობას ემატება ბოლოდან.

RR. სისტემაში ყოველი წარმოქმნილ პროცესი მდგომარეობაში მზადყოფნა მყოფი პროცესების მიმდევრობას ემატება ბოლოდან. მიმდევრობის ელემენტებისათვის პროცესორის გამოყოფა ხორციელდება გარკვეული დროითი შუალედებით (კვანტით), რომლის ამოწურვის შემდეგ, თუ დრო პროცესის დასასრულებლად არ აღმოჩნდა საკმარისი, მაშინ ის ჩამოერთმევა მიმდინარე პროცესს, პროცესი გადადის მიმდევრობის ბოლოში (თავიდან იკავებს რიგს) და პროცესორი შესასრულებლად გადაეცემა მიმდევრობის შემდეგ წევრს. თუ დროითი კვანტი საკმარისი აღმოჩნდა პროცესის დასასრულებლად, მაშინ პროცესორი გადაეცემა მიმდევრობის შემდეგ წევრს. პროცესორის გამოყოფა ციკლურად ხორციელდება მანამ მიმდევრობა შეიცავს ერთ მაინც წევრს.

SJF. ამ ალგორითმის გამოყენება გულისხმობს დამგეგმავის მიერ შესასრულებლად პირველი არჩეული იყოს მოკლე (ნაკლები პროცესორული დროის საჭიროების მქონე) ამოცანა. ამ ალგორითმშიც გამოიყენება ერთი მიმდევრობა. მიმდევრობა დალაგებულია ზრდადობით შესასრულებლად საჭირო დროითი შუალედის მიხედვით.

SJF ალგორითმში პრიორიტეტის დამატებით მიიღება განსხვავებული ალგორითმი. იგულისხმება, რომ პროცესების შესრულებისათვის საჭირო დრო წინასწარაა ცნობილი. სისტემაში ყოველი წარმოქმნილი პროცესის შესრულებისათვის საჭირო დრო ედრება მიმდინარე პროცესის დასრულდებისათვის საჭირო დროს. თუ ეს უკანასკნელი მეტია წარმოქმნილი პროცესის შესრულების დროზე, მაშინ მიმდინარე პროცესი წყვეტს შესრულებას და მის ადგილს იკავებს წარმოქმნილი პროცესი. თუკი წარმოქმნილი პროცესის შესრულების დრო მეტია მიმდინარე

პორცესის დასრულებისათვის საჭირო დროზე, მაშინ წარმოქმნილი პროცესი ემატება მდგომარეობაში მზადყოფნა მყოფი პროცესების მიმდევრობას და იქ იკავებს შესაბამის ადგილს (პრიორიტეტის მიხედვით). ასეთნაირად რეალიზებული SJF ალგორითმი იძლევა სისტემაში წარმოქმნილი მოკლე დროის საჭიროების მქონე პროცესების დროული შესრულების შესაძლებლობას.

4. კავშირის არხები

pipe. კავშირის არხებით პროცესებს შორის ინფორმაციის გადაცემის საკმაოდ მარტივ მეთოდს წარმოადგენს მონაცემების გადაცემა pipe-ით (არხი, მილი). წარმოვიდგინოთ, რომ გამოთვლით სისტემაში გვაქვს გარკვეული მილი, რომლის ერთი ბოლოდან პროცესებს შეუძლიათ ინფორმაციის „გადაღვრა“, ხოლო მეორედან კი შემოსული ნაკადის მიღება. ასეთი მეთოდი ახდენს ნაკადების შეტანა/გამოტანის მოდელის რეალიზებას. ოპერაციულ სისტემაში მილის განთავსების შესახებ ინფორმაციას ფლობს მხოლოდ მისი წარმომქმნელი პროცესი. ეს ინფორმაცია მის მიერ შეიძლება გაზიარებული იყოს მხოლოდ მემკვიდრე პროცესებთან, ანუ მშობელი - შვილი დამოკიდებულებით დაკავშირებულ პროცესებთან. შესაბამისად, კავშირისათვის pipe-ის გამოყენება შეუძლიათ მხოლოდ იმ პროცესებს, რომელთაც ყავთ კავშირის მოცემული არხის წარმოქმნელი “წინაპარი”.

fifo. თუ pipe-ის შემქმნელ პროცესს ექნებოდა სისტემაში სხვა პროცესებისათვის pipe-ის გაზიარების შესაძლებლობა, ანუ შექმნილი pipe-ის ხილულად გადაქცევის შესაძლებლობა, მაგალითად, ოპერაციულ სისტემაში მისი გარკვეული სახელით დარეგისტრირების გზით, მივიღებდით ობიექტს, რომელსაც FIFO ეწოდება. FIFO-ს გამოყენებით შესაძლებელია სისტემაში ნებისმიერ პროცესებს შორის კავშირის ორგანიზება.

კავშირის მიმართულება. კავშირი ერთმიმართულებიანია, თუ მასთან ასოცირებულ პროცესს კავშირის საშუალება შეუძლია გამოიყენოს მხოლოდ ერთი მიმართულებით ან ინფორმაციის მისაღებად, ან მის გადასაცემად. ორმიმართულებიანკავშირში ყოველ პროცესს, რომელიც მონაწილეობს ურთიერთქმედებაში, შეუძლია გამოიყენოს კავშირი მონაცემების მისაღებად და მის გადასაცემად. კომუნიკაციურ სისტემებში ერთმიმართულებიან კავშირს ეწოდება **სიმპლექსური**, ორ მიმართულებიანს კი მონაცემების სხვადასხვა მიმართულებით მიმდევრობით გადაცემის შესაძლებლობით - **ნახევრად-დუპლექსური**, ხოლო მონაცემების ერთდროული გადაცემის შესაძლებლობით კი **დუპლექსური**.

5. შეჯიბრის მდგომარეობა, კრიტიკული სექცია, ალგორითმები

შეჯიბრის მდგომარეობა. პროცესები შესაძლებელია შეთანხმებულად იყენებდნენ ოპერაციულ სისტემაში არსებულ მონაცემთა საერთო საცავს. ეს საცავი შეიძლება იყოს განთავსებული ოპერატიულ მეხსიერებაში ან წარმოდგენილი იყოს რაიმე ფაილის სახით.

კრიტიკული სექცია. პროგრამის იმ ნაწილს, რომელშიც გამოიყენება საერთო მეხსიერებაზე წვდომა კრიტიკული სექცია ეწოდება.

პროცესის მიერ შესრულებული სამუშაო, რომ იყოს მისაღები საჭიროა შემდეგი პირობების შესრულება:

1. ორ პროცესს არ შეუძლია ერთდროულად იმყოფებოდეს საკუთარ კრიტიკულ სექციაში;
2. არ უნდა არსებობდეს არანაირი წინასწარი მოსაზრება პროცესორების სავარაუდო რაოდენობაზე;
3. არცერთი პროცესი, რომელიც სრულდება საკუთარი კრიტიკული სექციის გარეთ არ შეიძლება იყოს ბლოკირებული სხვა პროცესის მიერ;
4. პროცესები უსასრულოდ არ უნდა ელოდებოდნენ საკუთარ კრიტიკულ სექციაში შესვლას.

წყვეტის აკრძალვა. წყვეტის აკრძალვის შემთხვევაში შეუძლებელი იქნება ტაიმერიდან წყვეტის განხორციელება და შესაბამისად პროცესორიც ვერ შეძლებს სხვა პროცესზე გადართვას. ამ შემთხვევაში არ არსებობს საფრთხე რომელიმე პროცესმა განახორციელოს საკუთარ კრიტიკულ სექციაში შესვლა, მაგრამ შეიძლება წაარმოიშვას სხვა ტიპის პრობლემა.

მრავალპროცესორული სისტემის შემთხვევაში წყვეტის აკრძალვა მოქმედებს მხოლოდ იმ პროცესორზე, რომელსაც შეეხება ბლოკირების ინსტრუქცია. სხვა დანარჩენი პროცესორები ჩვეულ რეჟიმში გააგრძელებენ მუშაობას. მიუხედავად იმისა, რომ მრავალპროცესორულ სისტემაში შესაძლებელია მეთოდმა გაამართლოს მაინც დაუშვებელია პროცესისათვის წყვეტის აკრძალვის შესაძლებლობის მიცემა.

ცვლადი-ბოქლომი. ამ მეთოდის გამოყენება გულისხმობს სისტემაში პროცესებისათვის საერთო ცვლადის (ცვლადი-ბოქლომი) შემოღებას, რომელიც გააკონტროლებს პროცესების შესვლას საკუთარ კრიტიკულ სექციაში. ცვლადი-ბოქლომის საწყისი ინიციალიზაცია ხდება 0-ვანი მნიშვნელობით, რაც პროცესებისათვის აღნიშნავს, რომ საკუთარ კრიტიკულ სექციაში არ იმყოფება არცერთი პროცესი და ნებისმიერ მათგანს სურვილის შემთხვევაში შეუძლია შევიდეს იქ. პროცესი საკუთარ კრიტიკულ სექციაში შესვლის შემდეგ ცვლის ცვლადი- ბოქლომის მნიშვნელობას 1-ით, რაც იმის მანიშნებელია, რომ ურთიერთქმედი პროცესებიდან ერთერთი იმყოფება საკუთარ კრიტიკულ სექციაში და, თუ რომელიმე პროცესი საჭიროებს საკუთარ კრიტიკულ სექციაში შესვლას, მაშინ მას მოუწევს დალოდება სანამ მიმდინარე პროცესი არ გამოვა იქიდან. პროცესი საკუთარი კრიტიკული სექციიდან გამოსვლის შემდეგ ცვლად-ბოქლომს უბრუნებს საწყის მნიშვნელობას (0-ს).

მკაცრი მიმდევრობა. კრიტიკული სექციის პრობლემის გადაწყვეტის შემდეგი მეთოდი გულისხმობს პროცესების წარმოდგენას მკაცრი მიმდევრობის სახით. პროცესები მიმდევრობით შედიან საკუთარ კრიტიკულ სექციაში. იქიდან გამოსვლის შემდეგ მიმდევრობის შემდეგ წევრს ეძლევა საკუთარ კრიტიკულ სექციაში შესვლის შესაძლებლობა.

6. ურთიერთბლოკირება, ურთიერთბლოკირების წარმოქმნის აუცილებელი და საკმარისი პირობა

ურთიერთბლოკირება. ხშირად პროცესები საკუთარი სამუშაოს შესასრულებლად საჭიროებენ ერთზე მეტ რესურსს. მაგალითად, განვიხილოთ სიტუაცია, რომლის დროსაც სისტემაში გვაქვს ორი პროცესი, P1, P2, და ორი რესურსი: სკანერი და დისკური მოწყობოლიბა. დავუშვათ, რომ ორივე პროცესი საჭიროებს თითოეულ რესურსს და რესურსების გამოყოფა ხორციელდება პროცესებისათვის თითოთითოდ. დავუთვათ, P1 პროცესი დაპროგრამირებულია ისე, რომ მან პირველი გამოიყენოს სკანერი და მეორე დისკური მოწყობილობა, ხოლო P2 პროცესი კი პირიქით. როგორც წესი, პირველი P1 პროცესს გამოეყოფა სკანერი, ხოლო P2 -ს დისკური მოწყობილობა. P1 პროცესი სკანერთან მუშაობის დასრულების და მისგან მონაცემების მიღების შემდეგ საჭიროებს მეორე რესურსს და აკეთებს შესაბამის მოთხოვნას რესურსის გამოყოფაზე. მაგრამ მისი დაკმაყოფილება შეუძლებელია, ვინაიდან დისკურ მოწყობილობას იკავებს P2 პროცესი. ანალოგურად, P2 პროცესი დისკურ მოწყობილობასთან მუშაობის დასრულების შემდეგ გააკეთებს მოთხოვნას სკანერზე და მისი დაკმაყოფილებაც ასევე შეუძლებელია. ასეთ შემთხვევაში ორივე პროცესი იქნება ბლოკირებული სანამ არ გამონთავისუფლდება მათ მიერ მოთხოვნილი რესურსი. წარმოქმნილ სიტუაციას ეწოდება **ურთიერთბლოკირება** (**deadlock**) **ან ჩიხი**.

ურთიერთბლოკირების წარმოქმნის აუცილებელი და საკმარისი პირობები. ურთიერთბლოკირების წარმოქმნისათვის აუცილებელია და საკმარისი შემდეგი 4 პირობის შესრულება:

1. **ურთიერთგამორიცხვის პირობა (mutual exclusion)**. დროის ნებისმიერ მომენტში რესურსი ან გამოყოფილია მხოლოდ ერთი პროცესისათვის ან თავისუფალია;
2. **რესურსის ლოდინის პირობა (hold and wait)**. პროცესს, რომელსაც დაკავებული აქვს გარკვეული რესურსი შეუძლია მოითხოვოს ახალი რესურსი;
3. **გაუნაწილებლობის პირობა (no preemption)**. პროცესისათვის მის მიერ დაკავებული რესურსის იძულებითი ჩამორთმევა შეუძლებელია. პროცესმა დაკავებული რესურსი უნდა გამოანთავისუფლოს საკუთარი სურვილით;
4. **ციკლური ლოდინის პირობა (circular wait)**. უნდა არსებობდეს ორი ან მეტი პროცესისაგან შემდგარი წრიული მიმდევრობა, რომელშიც მიმდევრობის ყოველი წევრი ელოდება მის შემდეგ მდგომი წევრის მიერ დაკავებული რესურსის გამონთავისუფლებას.

ჩამოყალიბებული პირობებიდან ერთერთის დარღვევის შემთხვევაში სისტემაში არ გვაქვს ურთიერთბლოკირების მდგომარეობა.

7. რესურსები

რესურსი შეიძლება იყო აპარატული (მყარი დისკი, პროცესორი, მეხსიერება) ან ინფორმაციული (ფაილები, მონაცემთა ბაზა). როგორც უკვე აღვნიშნეთ ყოველი რესურსი დროის ნებისმიერ მომენტში შესაძლებელია გამოეყოს მხოლოდ ერთ პროცესს. შესაბამისად, თუ რამდენიმე პროცესი საჭიროებს ერთიდაიმავე რესურსს მათ ამ რესურსის მიღება შეუძლიათ რიგრიგობით. თუ ყოველი რესურსი ოპერაციულ სისტემაში წარმოდგენილი იქნებოდა რამდენიმე ასლის სახით, მაშინ ერთი

კონკრეტული რესურსის მომთხოვნი რამდენიმე პროცესის დაკმაყოფილება იქნებოდა შესაძლებელი.

განაწილებადი. პროცესის მიერ დაკავებულ რესურსს, რომლის ჩამორთმევაც მისთვის შესაძლებელია „უმტკივნეულოდ“, მიეკუთვნება განაწილებად რესურსს. ამ ტიპის რესურსის მაგალითს წარმოადგენს მეხსიერება. სისტემაში არასაკმარის მეხსიერების არსებობის შემთხვევაში ხორციელდება პროცესის სრული ან ნაწილობრივი გადატანა მეხსიერების არიდან დისკზე, ხოლო მოგვიანებით კი მისი უკან დაბრუნება.

გაუნაწილებელი. პროცესისათვის გაუნაწილებელი რესურსის ჩამორთმევამ შესაძლებელია გამოიწვიოს პროცესის მიერ შესრულებული მნიშვნელოვანი სამუშაოს დაკარგვა. ურთიერთბლოკირებაში შესაძლებელია მონაწილეობდეს როგორც განაწილებადი, ისე გაუნაწილებელი რესურსი. რესურსების საგულდაგულო გადანაწილების ხარჯზე შესაძლებელია განაწილებადი რესურსებით გამოწვეული ურთიერთბლოკირების აცილება. ამიტომ ყურადღებას გავამახვილებთ მხოლოდ გაუნაწილებელი რესურსებით გამოწვეულ ურთიერთბლოკირებასთან გამკვლავების მეთოდებზე.

საზოგადოდ, სისტემაში რესურსების გამოყენება ხორციელდება მოვლენათა შემდეგი მიმდევრობით:

1. მოთხოვნა;
2. გამოყენება;
3. გამონთავისუფლება.

8. მეხსიერების ორგანიზაცია

იერარქიის თავში განთავსებულია ყველაზე სწრაფქმედი მეხსიერება, ნაკლები მოცულობითა და ინფორმაციის შესანახ ბიტზე მაღალი ღირებულებით, ხოლო იერარქიის ბოლო დონეზე განთავსებულია ნელი, იაფი ღირებულების და დიდი მოცულობის მეორადი მეხსიერების მოწყობილობები.

რეგისტრები - ეს არის სისტემაში გამოყენებული ყველაზე ძვირი და სწრაფქმედი მეხსიერება. ისინი დამზადებული იმავე ტექნოლოგიით რაც პროცესორები და სისწრაფითაც არ ჩამოუვარდებიან მას. შემდეგ დონეზე განთავსებულია ქეშ-მეხსიერება. მისი სისწრაფე იზომება დაყოვნებებით - დროითი შუალედით, რომელიც საჭიროა მონაცემების გადასაცემად. როგორც წესი, დაყოვნებები იზომება ნანოწამებში ან პროცესორის ციკლით.

იერაქიის შემდეგ დონეზე განთავსებულია ოპერატიული (ძირითადი, ფიზიკური) მეხსიერება. ოპერატიული მეხსიერება პროცესორისათვის მონაცემების გადასაცემად იყენებს პროცესორის სისწრაფესთან შედარებით დაბალი სისწრაფის სისტემურ სალტეს (system bus) და შესაბამისად დაყოვნება ამ შემთხვევაში მაღალია.

მეხსიერების შემდეგ დონეებზე განთავსებულია მეხსიერების მეორადი (გარე) მოწყობილობები. ელექტრული დისკები, მაგნიტური დისკები და მაგნიტური ლენტა წარმოადგენს გამოთვლით სისტემაში შენახვის დიდი მოცულობის და დაბალი ღირებულების მქონე მოწყობილობებს. ამ ტიპის მოწყობილობებში ინფორმაციაზე წვდომა ქეშ-მეხსიერებასთან შედარებით ხორციელდება რამდენიმე ასეულ ათასჯერ ნაკლები სისწრაფით. შენახვის მეორადი მოწყობილობების უპირატესობა მდგომარეობს იმაში, რომ ისინი არ არიან დამოკიდებული ელექტროენერგიაზე და უეცარი გამორთვის შემთხვევაში მათზე არსებული ინფორმაცია არ იკარგება. მეორე უპირატესობას წარმოადგენს ის, რომ მათ მონაცემების შესანახად გააჩნიათ დიდი მოცულობა.

9. მისამართების დაკავშირება

ლოგიკური მისამართების სივრცის მაქსიმალური მოცულობა განისაზღვრება პროცესორის თანრიგიანობით და გაცილებით აჭარბებს თანამედროვე სისტემებში ფიზიკური მისამართების სივრცის მოცულობას. შესაბამისად პროცესორს და ოპერაციულ სისტემას უნდა შეეძლოს პროგრამის კოდში მიმართვების (მისამართების) ასახვა რეალურ ფიზიკურ მისამართებზე, რომელიც შეესაბამება პროგრამის მიმდინარე მდებარეობას ძირითად მეხსიერებაში. მისამართების ასეთ ასახვას ეწოდება მისამართების ტრანსლირება ან მისამართების დაკავშირება.

ინსტრუქციებისა და მონაცემების დაკავშირება მეხსიერებასთან შესაძლებელია გაკეთდეს შემდეგ ნაბიჯებში:

- **კომპილაციის ეტაპზე** - როდესაც ცნობილია მეხსიერებაში პროცესის ზუსტიადგილმდებარეობა, მაშინ უშუალოდ გენერირდება ფიზიკური მისამართი. პროგრამისსაწყისი მისამართის შეცვლისას საჭიროა მისი კოდის ხელახალი კომპილირება.

- **ჩატვირთვის ეტაპი** - კომპილაციის ეტაპზე თუ უცნობია ინფორმაცია პროგრამის განთავსებაზე, კომპილატორი აგენერირებს გადატანად კოდს. ამ შემთხვევაში საბოლოო დაკავშირების გადადება ხდება ჩატვირთვის მომენტამდე. თუ საწყისი მისამართი იცვლება, საჭიროა მხოლოდ კოდის გადატვირთვა შეცვლილი სიდიდის გათვალისწინებით.

- **შესრულების ეტაპი** - თუ პროცესი შესაძლებელია გადაადგილებული იყოსშესრულების დროს მეხსიერების ერთი მიდამოდან მეორეში, დაკავშირება გადაიდებაშესრულების ეტაპამდე. აქ სასურველია სპეციალიზირებული მოწყობილობების,მაგალითად, გადაადგილების რეგისტრების, არსებობა. მათი მნიშვნელობა ემატებაპროცესის მიერ გენერირებულ ყოველ მისამართს. უმეტესი თანამედროვე ოპერაციულისისტემა ანხორციელებს მისამართების ტრანსლირებას შესრულების ეტაპზე, ამისათვისსპეციალური აპარატურული მექანიზმების გამოყენებით.

10. მეხსიერების მართვის სტრატეგიები (ჩატვირთვის, განთავსების, ჩანაცვლების)

მეხსიერების მართვის სტრატეგიები იქმნება ძირითადი მეხსიერების ოპტიმალური გამოყენების მიზნით. ამ სტრატეგიებს ყოფენ სამ ნაწილად:

1. ჩატვირთვის სტრატეგია;
2. განთავსების სტრატეგია;
3. ჩანაცვლების სტრატეგია.

ჩატვირთვის სტრატეგია განსაზღვრავს, თუ როდის უნდა განხორციელდეს ახალი პროგრამის მონაცემებისა და ინსტრუქციების ჩატვირთვა ოპერატიულ მეხსიერებაში. ეს სტრატეგია იყოფა ორ ნაწილად: ჩატვირთვა მოთხოვნის საფუძველზე და წინასწარი ჩატვირთვა. ბოლო პერიოდამდე გამოიყენებოდა სტრატეგია ჩატვირთვა მოთხოვნის საფუძველზე, რომელიც გულისხმობს ჩასატვირთი პროგრამის მონაცემების და ინსტრუქციების გადატანას შენახვის მეორადი მოწყობილობიდან ოპერატიულ მეხსიერებაში მას შემდეგ რაც მასზე გაკეთდება მოთხოვნა. ამ სტრატეგიისათვის უპირატესობის მინიჭების მიზეზი იყო გამოთვლით სისტემებში დიდი მოცულობის ოპერატიული მეხსიერების არარსებობა. ასეთ შემთხვევაში ოპერატიულ მეხსიერებაში სხვადასხვა პროგრამების მონაცემებისა და ინსტრუქციების განთავსება გამოიწვევს მის უსარგებლოდ დაკავებას. გამოთვლით სისტემებში დიდ მოცულობით ოპერატიული მეხსიერების გამოჩენამ შესაძლებელი გახადა წინასწარი ჩატვირთვის სტრატეგიის გამოყენება, რამაც საგრძნობლად გაზარდა პროცესორის მუშაობის სისწრაფე.

ოპერატიულ მეხსიერებაში პროგრამის მონაცემების და ინსტრუქციების განთავსების აუცილებლობისას განთავსების სტრატეგია განსაზღვრავს, თუ სად უნდა განხორციელდეს მათი განთავსება. განთავსების სტრატეგია იყოფა სამ ნაწილად: პირველი შესაფერისი, საუკეთესოდ შესაფერისი და ნაკლებად შესაფერისი. პირველი შესაფერისი სტრატეგიის მიხედვით ახალი პროგრამის მონაცემებისა და ინსტრუქციების განთავსება ხორციელდება მოცულობით შესაფერის პირველივე დანაყოფში. საუკეთესოდ შესაფერისი სტრატეგიის მიხედვით პროგრამის მონაცემებისა და ინსტრუქციების განთავსება ხორციელდება ძირითადი მეხსიერების ისეთ დანაყოფში, რომელშიც მათი განთავსების შემდეგ რჩება მცირემოცულობის ადგილი. ნაკლებად შესაფერისი სტრატეგიის შემთხვევაში პროგრამის მონაცემებისა და ინსტრუქციების განთავსება შესაძლებელია განხორციელდეს ძირითადი მეხსიერების ისეთ დანაყოფში, რომელშიც საკმარისი ადგილი იქნება ახალი პროგრამის მონაცემებისა და ინსტრუქციების განსათავსებლად. ოპერატიულ მეხსიერებაში მონაცემების განთავსებამდე ხორციელდება მისი დაყოფა გარკვეული მოცულობის ბლოკებად. თუ ბლოკები ფიქსირებული მოცულობისაა, მაშინ მათ ფურცლები ეწოდებათ. წინააღმდეგ შემთხვევაში მათ სეგმენტებს უწოდებენ.

თუ ძირითად მეხსიერებაში ახალი პროგრამის მონაცემების და ინსტრუქციების განსათავსებლად საკამარისი ადგილი არაა, მაშინ იქიდან უნდა მოხდეს გარკვეულიპროგრამის მონაცემების და ინსტრუქციების წაშლა. თუ რომელი მონაცემები და ინსტრუქციები უნდა წაიშალოს ოპერატიული მეხსიერებიდან განსაზღვრავს ჩანაცვლების სტრატეგია.

11. მეხსიერების დაყოფა (ფურცლისებრი, სეგმენტური, ფურცლისებრ-სეგმენტური)

ფურცლისებრი. თუ მონაცემთა ბლოკებს გააჩნიათ ფიქსირებული მოცულობა, მაშინ შესაბამის ბლოკებს ეწოდებათ მეხსიერების ფურცლები (pages), ხოლო სისტემას მეხსიერების ასეთი წესით ორგანიზაციით კი სისტემა მეხსიერების ფურცლისებრი ორგანიზაციით (paging).

სეგმენტური. თუ მონაცემთა ბლოკებს გააჩნიათ ცვლადი მოცულობა, მაშინ შესაბამის ბლოკებს ეწოდებათ სეგმენტები (segment), ხოლო სისტემას მეხსიერების ასეთი წესით ორგანიზაციით კი სისტემა მეხსიერების სეგმენტური ორგანიზაციით (segmentation).

ფურცლისებრ-სეგმენტური. ზოგიერთ სისტემებში შეჯერებულია მეხსიერების ფურცლისებრი და სეგმენტური მიდგომა. ასეთ სისტემებში მთლიანი მეხსიერება დაყოფილია სეგმენტებად, ხოლო სეგმენტები კი თავის შიგნით ფიქსირებული მოცულობის მქონე ფურცლებად და შესაბამის სისტემას ეწოდება სისტემა მეხსიერების სეგმენტურ-ფურცლისებრი ორგანიზაციით.

12. ფაილური სისტემა (ფაილი, დირექტორია, ფაილური დესკრიპტორი, ერთ- და მრავალდონიანი ფაილური სისტემა, ლინკი)

ფაილები. ფაილი წარმოადგენს მონაცემთა ნაკრებს, რომლებზეც შესაძლებელია გახორციელდეს შემდეგი ოპერაციები:

- შექმნა (create file) - ახალი ფაილის შექმნა;
- გახსნა (open file) - ფაილის გამზადება მასზე დასაშვები მოქმედებების (კითხვა ან ჩაწერა) განსახორციელებლად;
- დახურვა (close file) - ფაილის შემდგომ გახსნამდე მასზე დასაშვები მოქმედებების ბლოკირება;
- განადგურება (destroy file) - ფაილის წაშლა;
- კოპირება (copy file) - ფაილის შიგთავსის ასლის გადატანა სხვა ფაილში;
- ასახვა (list file) - ფაილის შიგთავსის მონიტორის ეკრანზე ან ფურცელზე გამოტანა.

მონაცემთა ცალკეული ელემენტებზე, რომლებიც შენახულია ფაილებში, შესაძლებელია შემდეგი ოპერაციების განხორციელება:

- კითხვა (read data) - ფაილიდან მონაცემების კითხვა პროცესის მეხსიერებაში;
- ჩაწერა (write data) - პროცესის მეხსიერებიდან მონაცემების ჩაწერა ფაილში;
- განახლება (update) - ფაილში არსებული მონაცემეთა ელემენტების რაოდენობის შეცვლა;
- ჩასმა (insert data) - ფაილში ახალი ელემენტების ჩამატება;
- წაშლა (delete data) - ფაილიდან მონაცემთა ელემენტების წაშლა.

ფაილები ხასიათდებიან შემდეგი მახასიათებლებით:

- მოცულობა (size of file) - ფაილში შენახული მონაცემების მოცულობა;

- ადგილმდებარეობა (location of file) - ფაილის ადგილმდებარეობა შესანახ მოწყობილობაზე ან ფაილური სისტემაში;
- დაშვების უფლებები (accessibility of file) - ფაილზე წვდომის უფლებები;
- ტიპი (type) - ფაილის ტიპი. მაგალითად შესრულებადი ფაილი პროცესისათვის შეიცავს შესასრულებელ ინსტრუქციებს;
- შეცვლადობა (volatility of file) - ფაილში შენახულ მონაცემებში ცვლილების შეტანის სიხშირე;
- აქტიურობა (activity of file) - დროის მოცემულ შუალედში ფაილის ჩანაწერებზე მიმართვის სიხშირე.

ფაილი შეიძლება შედგებოდეს ერთი ან რამდენიმე ჩანაწერისაგან. ფიზიკური ჩანაწერი ან ფიზიკური ბლოკი ეს არის შესანახ მოწყობილობაზე ჩაწერილი ან მისგან წაკითხული ინფორმაციის ერთეული. ლოგიკური ჩანაწერი ან ლოგიკური ბლოკი ეს არის მონაცემთა ნაკრები, რომელიც პროგრამების მიერ შეიძლება გაგებული იყოს როგორც ინფორმაციის ერთეული. თუ ფიზიკური ჩანაწერი შეიცავს ზუსტად ერთ ლოგიკურ ჩანაწერს, მაშინ ამბობენ, რომ ის შედგება ღია მონაცემებისაგან (unblocked records). წინააღმდეგ შემთხვევაში ამბობენ, რომ ჩანაწერი შედგება ჩაკეტილი მონაცემებისაგან (blocked records).

დირექტორიები. ფაილურ სისტემაში ფაილების ორგანიზებული განთავსებისა და სწრაფი წვდომისათვის გამოიყენება დირექტორიები. დირექტორია ეს არის ფაილი, რომელიც შეიცავს მასში შემავალ ფაილებსა და დირექტორიებზე ჩამონათვალს და ინფორმაციას ფაილურ სისტემაში მათი ლოგიკური ადგილმდებარეობის შესახებ. ჩვეულებრივი ფაილებისაგან განსხვავებით დირექტორიები არ ინახავენ მონაცემებს.

ველები, რომლებსაც შეიძლება შეიცავდეს დირექტორია

ველი	აღწერა
სახელი	ფაილის სახელის შემცველი სიმბოლური სტრიქონი
ადგილმდებარეობა	ფაილურ სისტემაში ფაილის განთავსების ლოგიკური მისამართი (მაგალითად, სრული ან მიმართებითი სახელი)
მოცულობა	ფაილის მიერ დაკავებული ბაიტების რაოდენობა
ტიპი	ფაილის დანიშნულების აღმნიშვნელი სიმბოლო (მაგალითად, მონაცემების ფაილი ან დირექტორია)
მიმართვის დრო	ფაილზე ბოლო მიმართვის დრო
შეცვლის დრო	ფაილში განხორციელებული ბოლო ცვლილების დრო
შექმნის დრო	ფაილის შექმნის დრო

ფაილური დესკრიპტორი. ფაილის გახსნის ოპერაციის შესრულებისას ოპერაციული სისტემა ეძებს ფაილზე ინფორმაციას და დირექტორიათა სტრუქტურაში პოულობს მის ადგილმდებარეობას. ერთსადაიმავე ფაილის მრავალჯერადი გახსნის თავიდან აცილების მიზნით სისტემა ოპერატიულ მეხსიერებაში გახსნილ ფაილზე ინფორმაციას ინახავს ე.წ. ღია ფაილების ცხრილში შესაბამისი ჩანაწერის გაკეთების გზით. ფაილის გახსნის ოპერაცია აბრუნებს არაუარყოფით რიცხვით მნიშვნელობას - ფაილურ დესკრიპტორს, რომელიც წარმოადგენს ღია ფაილების ცხრილში შესაბამის ფაილზე ჩანაწერის ნომერს. ფაილზე განსახორციელებელი ყოველი შემდეგი მოქმედება ხორციელდება ფაილური დესკრიპტორის გამოყენებით.

ღია ფაილების ცხრილი ასევე შეიცავს ფაილის მართვის ველებს. ამ ველებში ინახება ფაილის სამართავად გამოყენებადი ინფორმაცია ე.წ. ფაილის ატრიბუტები. ფაილის ატრიბუტები შეიძლება განსხვავდებოდნენ სისტემიდან სისტემამდე. მაგალითად, ის შეიძლება შეიცავდეს შემდეგ ინფორმაციას:

- ფაილის სიმბოლური სახელი;
- ფაილის ადგილმდებარეობა შესანახ მოწყობილობაზე;
- საორგანიზაციო სტრუქტურა (მაგალითად, მიმდევრობითი ან ნებისმიერი დაშვების ფაილი);
- ინფორმაცია შესანახ მოწყობილობაზე;
- ინფორმაცია ფაილის ტიპზე;
- ინფორმაცია ფაილის ხასიათზე (დროებითი თუ მუდმივი);
- ფაილის შექმნის დრო და თარიღი;
- ფაილზე მიმართვების მთვლელი (მაგალითად, ფაილზე განხორციელებული კითხვის ოპერაციების რაოდენობა);

ერთდონიანი ფაილური სისტემები. ფაილური სისტემის ყველაზე მარტივ რეალიზაციას წარმოადგენს ერთდონიანი ანუ ბრტყელი ფაილური სისტემა. ასეთ სისტემებში ყველა ფაილი ინახება ერთ დირექტოირაში. დირექტორიის შიგნით ფაილების სახელები უნდა იყოს უნიკალური. ვინაიდან სხვადასხვა პროგრამების მიერ გამოყენებული ფაილების სახელები (მონაცემები განსხვავებული) შესაძლებელია იყვენენ იდენტური, ამიტომ ამ ტიპის ფაილურმა სისტემამ ვერ ჰპოვა ფართოდ გავრცელება.

იერარქიულად სტრუქტურირებული ფაილური სისტემები. საბაზო (root) დირექტორია განთავსებულია იერარქიის თავში და აღნიშნავს ფაილური სისტემის დასაწყისს.

ფაილის სახელი უნიკალური უნდა იყოს მხოლოდ მომხმარებლის დირექტორიის შიგნით. იერარქიულად სტრუქტურირებულ ფაილურ სისტემაში ყოველ დირექტორიას შეიძლება გააჩნდეს მრავალი ქვედირექტორია, მაგრამ მხოლოდ ერთი მშობელი დირექტორია. ფაილის სრულ სახელს ფაილურ სისტემაში წარმოადგენს გზა საბაზო დირექტორიიდან დანიშნულების ფაილამდე.

ლინკი (link). ლინკი არის დირექტორიაში განთავსებული ჩანაწერი, რომელიც უთითებს სხვა დირექტორიაზე ან დირექტორიაში განთავსებულ ფაილზე. მომხმარებელი სისტემაში ნავიგაციის გამარტივების მიზნით ხშირად იყენებს ლინკს. სისტემაში ლინკი შეიძლება არსებობდეს ორი ტიპის: ლოგიკური და ფიზიკური.

ლოგიკური ლინკი წარმოადგენს დირექტორიაში არსებულ ჩანაწერს, რომელიც უთითებს სხვა დირექტორიაში განთავსებულ ფაილამდე გზას. ფაილური სისტემა მითითებულ ფაილამდე გზას იკვლევს ლინკში არსებული მისამართით.

ფიზიკური ლინკი კი ეს არის დირექტორიაში არსებული ჩანაწერი, რომელიც უთითებს შესანახ მოწყობილობაზე ფაილის განთავსების ფიზიკურ მისამართს. ამ შემთხვევაში ფაილური სისტემა ფაილს პოულობს მისი ფიზიკური მისამართით.

13. ფაილის ორგანიზაციის მეთოდები (მიმდევრობითი, ინდექსირებული მიმდევრობა)

ფაილების ორგანიზაცია ეწოდება შესანახ მოწყობილობაზე მათი განთავსების წესს.

მიმდევრობითი (sequential) - შესანახ მოწყობილობაზე ჩანაწერის განთავსება ხორციელდება მათი ფიზიკური მიმდევრობით. ასეთი ორგანიზაცია დამახასიათებელია მაგნიტურ ლენტაზე შენახული ფაილებისათვის, ანუ მიმდევრობითი დაშვების მოწყობილობებისათვის;

ინდექსირებული მიმდევრობა (indexed sequential) - შესანახ მოწყობილობაზე ჩანაწერები თავსდება ლოგიკური მიმდევრობით, გასაღების გამოყენებით, რომელიც ინახება თითოეულ ჩანაწერში. სისტემაში მხარდაჭერილია ინდექსები, რომლებით უთითებენ გარკვეულ ძირითად ჩანაწერებს. ინდექსირებული მიმდევრობის ჩანაწერებზე დაშვება შესაძლებელია მათი გასაღებების მიმდევრობით, ან პირდაპირ სისტემაში შექმნილი ინდექსების მეშვეობით;