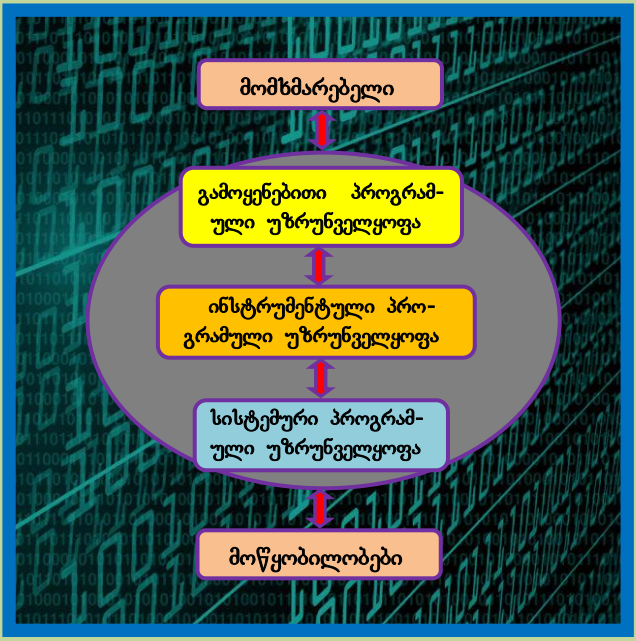
ალექსანდრე დუნდუა

კომპიუტერული სისტემების პროგრამული საშუალებები (SOFTWARE)



საგამომცემლო სახლი
„ტექნიკური უნივერსიტეტი“

პროგრამული უზრუნველყოფის მიმწოდებლები ცდილობენ საკუთარი პროდუქტი რაც შეიძლება ადვილად გამოსაყენებელი გახადონ. მათიმცდელობა დღემდე მთავრდება მხოლოდ დოკუმენტაციის გარეკანზე წარწერით: „ადვილად გამოსაყენებელი“.

Bill Gates

არსებობს პროგრამული უზრუნველყოფის შექმნის ორი მეთოდი: იგი უნდა იყოს იმდენად მარტივი, რომ ნათლად ჩანდეს ხარვეზების არარსებობა ან იმდენად რთული, რომ არ ჩანდეს არსებული ხარვეზები.

Tony Hoare

AlexanderDunduaAssociateProfessoroftheGeorgian TechnicalUniversity

საქართველოს ტექნიკური უნივერსიტეტი

ალექსანდრე დუნდუა

კომპიუტერულისისტემების პროგრამული საშუალებები (SOFTWARE)

რეკომენდებულია საქართველოს
ტექნიკური უნივერსიტეტის
სარედაქციო-საგამომცემლო საბჭოს
მიერ. 05.07.2019, ოქმი №2

თბილისი
2019

Uაკ 004.4

დამხმარე სახელმძღვანელო შეიცავს კომპიუტერული სისტემების პროგრამული საშუალებების ფორმირებისა და განვითარების საკვანძო საკითხებს, რომელთა შესწავლა გათვალისწინებულია სასწავლო დისციპლინით „კომპიუტერული სისტემები და გამოყენებითი ტექნოლოგიები“. მასში გადმოცემულია კომპიუტერული სისტემების პროგრამული უზრუნველყოფის სახეები, კლასიფიკაცია, აგებისა და ფუნქციონირების საფუძვლები. დიდი ადგილი აქვს დათმობილი ტექსტების დამუშავებისა და ცხრილური გამოთვლების ტექნოლოგიათა საკვანძო საკითხების განხილვას.

წიგნი განკუთვნილია სატრანსპორტო და მანქანათმშენებლობის ფაკულტეტის „ტრანსპორტის“, „საგზაო ინჟინერიისა“ და „ბიზნესის ორგანიზაცი-ისა და მართვის“ საგანმანათლებლო პროგრამათა ბაკალავრებისათვის. შეიძლება გამოყენებული იქნეს ტექნიკური პროფილის მაგისტრანტების, სპეციალისტებისა და კომპიუტერული სისტემებით დაინტერესებულ ადა-მიანთა ფართო წრისათვის.

რეცენზენტები: საქართველოს ტექნიკური უნივერსიტეტის
ენერგეტიკისა და ტელეკომუნიკაციის
ფაკულტეტის პროფესორი სერგო დადუნაშვილი,

საქართველოს ტექნიკური უნივერსიტეტის
სატრანსპორტო და მანქანათმშენებლობის
ფაკულტეტის ასოცირებული პროფესორი მურთაზ
პაპასკირი

ISBN 978-9941-28-514-1

http://www.gtu.ge

Verba volant,
scripta manent

ს ა რ ჩ ე ვ ი

I თავი
პროგრამული უზრუნველყოფის ფუნქციონირების საფუძვლები

1.1. ზოგადი ცნობები

პროგრამული უზრუნველყოფა ეწოდება კომპიუტერული სისტემის ფუნქციონირებისათვის საჭირო კომპიუტერული პროგრამების, პროცედურებისა და სათანადო დოკუმენტაციების ერთობლიობას (*IEEE Std 829—2008*) ანუ, მოკლედ, კომპიუტერის მართვისათვის გამოყენებულ პროგრამათა ერთობლიობას (პროგრამას)(*IEEE Std 829-2008*). მოყვანილი განმარტებების თანახმად კომპიუტერული სისტემის პროგრამულ უზრუნველყოფას აქვს რთული სტრუქტურა (ცხრილი ***1.1***), რომლის მსხვილი სტრუქტურული ერთეულებია სისტემური პროგრამული უზრუნველყოფა, ინსტრუმენტული პროგრამული უზრუნველყოფა და გამოყენებითი პროგრამული უზრუნველყოფა.

1. სისტემური პროგრამული უზრუნველყოფა

სისტემური პროგრამული უზრუნველყოფა წარმოადგენს პროგრამებისა და პროგრამათა კომპლექსების ერთობლიობას, რომლებიც უზრუნველყოფს კომპიუტერისა და კომპიუტერული ქსელების მუშაობას. მისი დანიშნულებაა: ***ა)*** სხვა პროგრამებისათვის შექმნას ფუნქციონირებისათვის საჭირო ოპერაციული გარემო; ***ბ)*** სა-იმედო და ეფექტური გახადოს კომპიუტერისა და კომპიუტერული ქსლის მუშაობა; ***გ)*** მოახდინოს კომპიუტერისა და კომპიუტერული ქსლის აპარატურის დიაგნოსტირება და პროფილაქტიკა; ***დ)*** შეასრულოს დამხმარე ტექნოლოგიური პროცესები (ფაილების კოპირება, დაარქივება, პროგრამებისა და მონაცემთა ბაზები, ფაილების აღდგენა და ა.შ.).

სისტემური პროგრამული უზრუნველყოფა მჭიდროდაა დაკავშირებული კომპიუტერთან და მის განუყოფელ ნაწილს წარმოადგენს. იგი შედგება (იხ. ცხრილი ***1.1***): ***საბაზისო, სერვისული*** და ***ტესტური*** პროგრამული უზრუნველყოფებისაგან. მოკლედ განვიხილოთ თითოეული მათგანი.

ცხრ.1.1. პროგრამული უზრუნველყოფის სტუქტურა

პროგ. უზრუნველყოფის:		შენიშვნები
საზეები	ქვესაზეები	
I. სისტემური პროგრამული უზრუნველყოფა (პუ)	I.1. საბაზისო პუ	ოპერაციული სისტემები და მათი გარსაცმები
	I.2სერვისული პუ	1) ანტივირუსული პროგრამები; 2) ფაილების, საქაღალდეების, დისკების არქივატორები; 3) სარეზერვო კოპირების უტილიტები; 4) დისკებისSMART-რევიზორები; 5)ანტიფიშინგები, რუტკიტებისაგან დაცვის უტილიტები და ა. შ.
	I.3. ტესტური პუ	უწესივრობათა აღმოჩენა-გასწორებისათვის
II. ინსტრუმენტული პროგრამული უზრუნრუნველყოფა	II.1. დაპროგრამების ენები	დაპროგრამების დაბალი და მაღალი დონის ენები
	II.2. დაპროგრამების სისტემები	1) ტრანსლატორები; 2) პროგრამების დამუშავების გარემო; 3) საცნობარო პროგრამების ბიბლიოთეკები, 4) გამმართველები; 5) კავშირების რედაქტორები და ა. შ.
	II.3. CASS-საშუალებები	პროგრამების ავტომატიზებულად შემქმნელი პროგრამული უზრუნველყოფა
III. გამოყენებითი პროგრამული უზრუნველყოფა	III.1. საბაზისო საინფორმაციო ტექნოლოგიებისპუ	1) ტექსტური რედაქტორები და პროცესორები; 2) ცხრილური პროცესორები; 3) მონაცემთა ბაზების სისტემები; 4) ოფისური პაკეტები; 5) გრაფიკული პროცესორები; 6) საპრეზენტაციო რედაქტორები; 7) საგამომცემლო სისტემები და ა. შ.
	III.2. მეთოდურად ორიენტირებულიპუ	*1)*მათემატიკური დაპროგრამებისა და ანალიზის, აგრეთვე სტატისტიკური ანალიზის ამოცანების გადწყვეტა; *2)* პროექტების მართვა; *3)* დისტანციური მართვის სისტემის საგანმანათლებლო მასალების დამუშავება; *4)* ბიზნეს-პროცესების სისტემათა მოდელირება; *5)* საექსპერტო სისტემების გარსაცმებად და ა. შ.
	III.3. ფუნქციურად ორიენტირებული პუ	სხვადასხვა საგნობრივ სფეროში მართვის ამოცანების პროგრამული რეალიზებისათვის განკუთვნილი პროგრამული უზრუნველყოფა

I.1.საბაზისო პროგრამული უზრუნველყოფა კომპიუტერში ახდენს ინფორმაციის დამუშავების პროცესის ორგანიზებას და გამოყენებითი პროგრამებისთვის ქმნის ნორმალურად ფუნქციონირებისათვის საჭირო გარემოს. მათ მიეკუთვნება **ოპერაციული სისტემები და მათი გარსაცმები.**

I.2. სერვისული პროგრამული უზრუნველყოფა ეწოდება საბაზისო პროგრამული უზრუნველყოფის შესაძლებლობების გამაფართოვებელი პროგრამებისა და პროგრამული კომპლექსების ერთობლიობას. მასში გაერთიანებულია: ***ა)***ანტივირუსული პროგრამები; ფაილების, საქაღალდეებისა და დისკების არქივატორები; ***ბ)*** სარეზერვო კოპირების უტილიტები, დისკების ***SMART***-რევიზორებად წოდებული უტილიტები; **გ)** ანტიფიშინგები, რუტკიტებისაგან დაცვის უტილიტები და ა. შ.

*გავეცნოთ სერვისულ პროგრამულ უზრუნველყოფაში შემავალ **პროგრამა-უტილიტებს**, რომლებიც **I.1** ცხრილშია ჩამოთვლილი. **უტილიტა** არის სისტემის მომსახურებისა და მუშაობის ოპტიმზებისათვის განკუთვნილი დამხმარე პროგრამა, რომელიც წყვეტს ამოცანებს, რომელთა გადაწყვეტა თავად სისტემას არ შეუძლია. ასეთი პროგრამების უმრავლესობა განკუთვნილია ფაილური სისტემისა და დისკების მომსახურებისათვის, თუმცა არსებობს კომპიუტერული ვირუსებისაგან სისტემის დამცავი უტილიტებიც. თავდაპირველად **უტილიტა** პატარა პროგრამას წარმოადგენდა. დღეისათვის უტილიტების მოცულობა რამდენიმე ათეულ მეგაბაიტს აღწევს და სირთულით მხოლოდ საოფისე პაკეტებს ჩამოუვარდება. ადრე თუ მას მხოლოდ ერთი ან ორი ოპერაციის შესრულება შეეძლო, ახლა იგი გაცილებით მეტ ოპერაციებს ასრულებს.*

■ ***ანტივირუსული პროგრამა*** *ეწოდება სპეციალიზებულ პროგრამას, რომელიც განკუთვნილია, კერძოდ, კომპიუტერული ვირუსების, ზოგადად კი მავნედ წოდებულიარასასურველი პროგრამების აღმოსაჩენად და ასეთი პროგრამებით „დაინფიცირებული“ (მოდიფიცირებული) ფაილების აღსადგენად, ავრეთვე პროფილაქტიკისათვის – „დაინფიცირებული ფაილების ან ოპერაციული სისტემის აღსადგენად. **კომპიუტერული ვირუსი** მავნე პროგრამული უზრუნველყოფის სახეობაა, რომელსაც შეუძლია შექმნას საკუთარი თავის ასლი და ჩანერგოს იგი სხვა პროგრამების **კოდში,** სადაც ტვირთო სექტორებში, ავრეთვე საკუთარი თავის ასლი გაავრცელოს კავშირის არხებით. **კოდი** ეწოდება რაიმე სასრული ალფაბეტის სიმბოლოების სასრული მოწესრიგებული სიმრავლის ურთიერთცალსახა ასახვას ინფორმაციის*

გადაცემის, შენახვის ან გარდაქმნის კოდირებისათვის გამოსაყენებელი სიმბოლოების, როგორც წესი, უფრო ფართო სიმრავლეზე, რომელიც სავალდებულო არაა იყოს მოწესრიგებული.

■ ***არქივაცია ეწოდება*** *მეხსიერების ეკონომიის მიზნით ერთი ან რამდენიმე ფაილში არსებული მონაცემების შეკუმშვასა და ერთ* ***საარქივო ფაილში*** *ამ შეკუმშული მონაცემების განთავსებას. მონაცემების არქივაცია მნიშვნელოვანი დანაკარგების გარეშე იმ ფაილების ფიზიკური ზომების შემცირებაა, რომლებშიც შეინახება მონაცემები.* ***არქივატორი*** *ეწოდება პროგრამას (პროგრამის კომპლექსს), რომელსაც შეუძლია როგორც შეკუმშოს ფაილი, ისე შეკუმშული ფაილი დააბრუნოს საწყის მდგომარეობაში (აღადგინოს ივი. არქივატორებს ზოგჯერ* ***ჩამლაგებელ (შემფუთავ) უტილიტებსა და სამომხმარებლო პროგრამებსაც*** *უწოდებენ, რომლებიც საშუალებას გვაძლევს ფაილის ასლი შეკუმშული სახით მოვათავსოთ საარქივო ფაილში.*

■ ***სარეზერვო კოპირების უტილიტა*** *წარმოადგენს პროგრამას, რომელიც სხვა მზიდზე ქმნის საწყისი ინფორმაციის ასლს (***Backu.***) და შეუძლია ასლიდან საწყისი მონაცემების აღდგენა. ასეთი უტილიტაა, მაგალითად,* ***Handy Backup.***

■ ***დისკების SMART-რევიზორებად წოდებული უტილიტები****. ინგლისური აბრევიატურა* ***SMART*** *(****S****elf-****M****onitoring,****A****nalysis and* ***R****eporting* ***T****echnology) გაიშიფრება როგორც „თვითკონტროლის, ანალიზისა და ანგარიშგების ტექნოლოგია”.*

აღნიშნული უტილიტა გამოიყენება თვითდიაგნოსტირების ჩაშენებული აპარატურის მეშვეობით ხისტი მდგომარეობის შეფასებისა და მისი მწყობრიდან გამოსვლის წინასწარმეტყველებისათვის. აღნიშნული ტექნოლოგია ***ATA****-სა და მისი უფრო თანამედროვე* ***SATA****-ს პროტოკოლის ნაწილია.* ***ATA*** *(****A****dvanced* ***T****echnology* ***A****ttachment– მოწინავე გამოყენებითი ტექნოლოგია) წარმოადგენს კომპიუტერთან დამგროვებლის (ხისტი დისკისა და ოპტიკური დისკოსატარების) მიერთების პარალელურ ინტერფეისს.* ***SATA*** *(****S****erial* ***ATA****– მიმდევრობითი* ***ATA****) არის* ***ATA*** *ინტერფეისის განვითარება რომელიც ახდენს ინფორმაციის მიმდევრობით გაცვლას. მისი გამოჩენის შემდეგ* ***ATA****-ძ მიიღო სახელწოდება* ***PATA*** *(****P****arallel* ***ATA****).*

დღეისათვის ფართოდ გავრცელებული ***USB*** *„ფლეშკებში“ ხშირად ვერ გამოიყენება* ***SMART****-რევიზორები, რადგან ისინი სულ სხვა პროტოკოლზეა დაფუძნებული. არსებობს* ***SAT*** *სპეციფიკის შესაბამისად მომუშავე* ***SATA-USB****გადამყვანი, რომელთაგანაც ზოგიერთი მათგანი აღნიშნული უტილიტის გამოყენების საშუალებას გვაძლევს.*

■ ***რუტკიტებისაგან თავდაცვის უტილიტები. რუტკიტი*** *(****Rootkit****) არის სხვადასხვა გზით (ინტერნეტიდან მიღებული პროგრამიდან ან წერილის*

ფაილიდან) კომპიუტერში შეღწევის უნარის მქონე ***მავნე პროგრამა****. კომპიუტერის მომხმარებლის მიერ ჩატარებული მანიპულაციის მეშვეობით რუტკიტი გააქტიურდება და შეიძენს კომპიუტერის მართვის უნარს. ანტივირუსულ პროგრამებს შეუძლია რუტკიტი კომპიუტერში შეღწევის ეტაპზე „გამოიჭიროს“ და გააუვნებელყოს. ამ დროს რუტკიტი თუ არ იქნა გაუვნებელყოფილი და გააქტიურდა, მაშინ ანტივირუსი დაკარგავს მისი აღმოჩენის უნარს. გააქტიურებული რუტკიტის მეშვეობით ჰაკერს (მავნე პროგრამის გამავრცელებელ პიროვნებას) შეუძლია კომპიუტერიდან მიიღოს მისთვის საჭირო ნებისმიერი ინფორმაცია, მართოს კომპიუტერი და შეასრულოს სხვადასხვა თაღლითური ქმედებები.*

რუტკიტებისაგან თავდაცვის უტილიტები საშუალებას გვაძლევს ვებრძოლოთ რუტკიტებს. ასეთი უტილიტას ერთ-ერთი სახეა Kaspersky Virus Removal Tool 2015.

I.3. ტესტური პროგრამული უზრუნველყოფა განკუთვნილია კომპიუტერისა და კომპიუტერის ქსლის მუშაობის პროცესში წარმოშობილი შეცდომების დიაგნოსტირებისა და გასწორებისათვის.

2. ინსტრუმენტალი პროგრამული უზრუნველყოფა

ინსტრუმენტული პროგრამული უზრუნველყოფა ეწოდება ახალი პროგრამული პროდუქტის შექმნის, გამართვისა და დანერგვისათვის საჭირო პროგრამებისა და პროგრამული საშუალებების ერთობლიობას. მას მიეკუთვნება (იხ. ცხრილი ***1.1***) დაპროგრამების ენები და სისტემები, აგრეთვე პროგრამების შექმნის პროცესის ავტომატიზებისათვის განკუთვნილი ***CASS*** (***C****omputer* ***A****dded* ***S****oftware* ***E****ngineering*)-საშუალებები. მოკლედ განვიხილოთ თითოეული მათგანი.

II.1.დაპროგრამების ენა წარმოადგენს კომპიუტერული პროგრამების ჩასაწერად განკუთვნილ ფორმალურ ენას.

ზემოთ მოყვანილ განმარტებაში გამოყენებული ***ფორმალური ენა*** ეწოდება სასრული ალფაბეტათი შედგენილი სასრული რაოდენობის სიტყვების სიმრავლეს, ხოლო ***კომპიუტერული პროგრამა*** – კომპიუტერული ინსტრუქციებისა და მონაცემების კომბინაციას, რომელიც გამოთვლითი ტექნიკის აპარატურულ უზრუნველყოფას სჭირდება გამოთვლებისათვის ან მართვის ფუნქციების შესასრულებლად.

დაპროგრამების ენა განსაზღვრავს ლექსიკურ, სინტაქსურ და სემანტიკურ წესებს, რომლებიც საჭიროა პროგრამის გარე სახის

ჩამოყალიბებისა და კომპიუტერის მიერ იმ მოქმედებების შესასრულებლად, რომლებსაც განსაზღვრავს დაპროგრამების ენა.

II.2.დაპროგრამების სისტემის მხარდაჭერით სრულდება პროგრამების შესაქმნელად საჭირო სამუშაოს ყველა ეტაპი; კერძოდ, დაპროგრამების ენაზე ჩაიწერება საწყისი კოდი; შესრულდება კომპილაციის, ინტერპრეტაციისა, დოკუმენტირებისა და თანმხლები პროგრამული პროდუქტების ფორმირების პროცესები. მისი ერთ-ერთი ძირითადი კომპონენტია ***დაპროგრამების ენის ტრანსლატორი.*** იგი სპეციალურ პროგრამაა, რომელიც დაპროგრამების ენაზე დაწერილ ტექსტს გადათარგმნის კონკრეტული კომპიუტერის სამანქანო ენაზე; თარგმინის გამოყენებულ ხერხზე დამოკიდებულებით ტრანსლატორბი იყოფა ***კომპილატორებად*** და ***ინტერპრეტატორებად.***

კომპილატორი *საწყის ენაზე დაწერილი პროგრამის ყველა სიტყვას სათითაოდ გადათარგმნის სამანქანო (ორობით) ენაზე, მიღებული სიტყვების კომპილაციის გზით წარმოქმნის სამანქანო ენაზე დაწერილ პროგრამას, რომელიც შეინახება დისკზე. კომპიუტერს მხოლოდ დისკზე შენახული ასეთი პროგრამის შესრულება შეუძლია.*

ინტერპრეტატორი *სამანქანო ენაზე თარგმნილი თითოეულ სიტყვის შესაბამის ბრძანებას კომპიუტერი დაუყოვნებლივ ასრულებს: თარგმნილი ბრძანებები (სიტყვები) არ კომპილირდება და დისკზე არ შეინახება.*

II.3.CASS (***C****omputrt* ***A****dded* ***S****oftware* ***E****ngineering*) ***-საშუალებები*** (იხ. ცხრილი ***1.1***) პროგრამების შექმნის პროცესის ავტომატიზებისათვის გამოიყენება.

3. გამოყენებითი პროგრამული უზრუნველყოფა

გამოყენებითი პროგრამული უზრუნველყოფა ეწოდება კონკრეტულ საგნობრივ სფეროში გარკვეული კლასის ამოცანების გადასაწყვეტად განკუთვნილი ურთიერთდაკავშირებული პროგრამების კომპლექსს.

გამოყენებითი პროგრამული უზრუნველყოფა მიეკუთვნება უშუალოდ მომხმარებლისათვის განკუთვნილი პროგრამული პროდუქტების უფართოეს კლასს. იგი (იხ. ცხრილი ***1.1***) მოიცავს საბაზისო საინფორმაციო ტექნოლოგიებისათვის განკუთვნილ, აგრეთვე მეთოდურად ორიენტირებულ და ფუნქციურად ორიენტირებულ (საგნობრივ) პროგრამულ უზრუნველყოფებს. მოკლედ განვიხილოთ თითოეული მათგანი.

III.1. საბაზისო საინფორმაციო ტექნოლოგიების პროგრამული უზრუნველყოფაში გაერთიანებულია სხვადასხვა საინფორმაციო სისტემის ურთიერთგანსხვავებული კლასის ამოცანების გადასაწყვეტად განკუთვნილი ფართო სპექტრის პროგრამები. ისინი პირობითად შეიძლება დავყოთ (იხ. ცხრ. ***1.1***): ***ა)*** ტექსტურ რედაქტორებად და პროცესორებად; ***ბ)*** ცხრილურ პროცესორებად; ***გ)*** მონაცემთა ბაზების მმართველ სისტემებად; ***დ)*** საოფისე ტიპის ინტეგრირებულ პაკეტებად; ***ე)*** გრაფიკულ პროცესორებად; საპრეზენტაციო რედაქტორებად; ***ვ)*** საგამომცემლო სისტემებად და ა. შ.

III.2. მეთოდურად ორიენტირებული პროგრამული უზრუნველყოფა საშუალებას გვაძლევს, ამოცანების გადასაწყვეტი მეთოდები და მოდელები გამოვიყენოთ ამ ამოცანების წარმომშობი საგნობრივი სფეროსაგან დამოუკიდებლად. მის მიერ გადასაწყვეტი საკითხების ნაწილი ***1.1*** ცხრილშია ჩამოთვლილი.

III.3. ფუნქციურად ორიენტირებული (საგნობრივი) პროგრამული უზრუნველყოფა გამოიყენება სხვადასხვა საგნობრივი სფეროს მმართველობითი ამოცანების პროგრამული რეალიზებისათვის. მაგალითად, სამრეწველო წარმოების, სავაჭრო ორგანიზაციის, საგანმანათლებლო დაწესებულების და ა. შ. მართვის ავტომატიზებული სისტემებისათვის.

4. დრაივერი

დრაივერი (ინგ. *driver* – მძღოლი) ეწოდება კომპიუტერულ პროგრამულ უზრუნველყოფას, რომლის მეშვეობითაც სხვა პროგრამული უზრუნველყოფას (ოპერაციული სისტემას) გარკვეული მოწყობილობის აპარატურულ მოწყობილობასთან შეღწევის შესაძლებლობა ეძლევა. ოპერაციული სისტემის ყიდვისას ჩვენ მასთან ერთად გვეწოდება კომპიუტერული სისტემის მუშაობისათვის აუცილებელი აპარატურული მოწყობილობების საკვანძო კომპონენტების დრაივერები. ოღონდ ზოგიერთი ისეთი მოწყობილობებისათვის, როგორიცაა ვიდეობარათი ან პრინტერი, შეიძლება დაგვჭირდეს ჩვეულებრივ ამ მოწყობილობების მწარმოებლების მიერ დამუშავებული სპეციალური დრაივერები.

ზოგადად სავალდებულო არ არის დრაივერი ურთიერთზემოქმედებდეს აპარატურულ მოწყობილობასთან, მას შეუძლია მათი მხოლოდ იმიტირება (ასეთია, მაგალითად, პრინტერის დრაივერი, რომელსაც ჩანაწერი პროგრამიდან ფაილში გამოაქვს), მოწყობილობების მართ-

ვასთან დაუკავშირებელი პროგრამული სერვისების მიწოდება და ა. შ.

1.2. ოპერაციული სისტემის დანიშნულება და ორგანიზების ძირითადი პრინციპები

ლიტერატურაში არ არსებობს ოპერაციული სისტემის საყოველთაოდ მიღებული და კომპაქტური განსაზღვრება. ბრიტანელმა მეცნიერმა ***დ. უ. ბარონმა*** (***D. W. Barron, 1935-2012***) წინა საუკუნის ***70***-იან წლებში, როდესაც ოპერაციული სისტემების გამოირჩეოდა დიდი მრავალფეროვნებით, ოპერაციული სისტემის შესახებ მოხდენილად შენიშნა: „***მე არ ვიცი რა არის ივი, მაგრამ მას ვცნობ დანახვისთანავე***“. ამის შემდეგ მდგომარეობა მნიშვნელოვნად არ შეცვლილა. ერთი მხრივ, „საერთო დანიშნულების“ სისტემები – ***Unix, Windows XP, z/OS*** ერთმანეთს ისე ჰგავს, რომ საქმე ანეგდოტურიც ხდება:***X-Ope*** კონსორციუმა ***1998*** წელს ***OS/390*** სისტემას ჩაუტარა ტესტირება, რომელმაც გვაჩვენა, რომ მას შეიძლებოდა მინიჭებოდა ***Unix***–ის სახელი; ამასთანავე, ჩამოითვლება ოპერაციულ სისტემებად წოდებული ათზე არანაკლები პროგრამა, რომლებსაც ერთმანეთთან ძალიან ცოტა რამ აქვს საერთო, ხოლო ***PIC*** მიკროკონტროლერისათვის არსებობს ***1,5*** გვერდიანი ასემბლერული ლისტინგი ამაყი სახელწოდებით ***„რეალური დროის ოპერაციული სისტემა“***. მხედველობაში თუ მივიღებთ ***1993*** წელს წამოწყებულ „შეტევას“ ოპერაციულ სისტემად ***Windows 95/98/ME***-ების არმიჩნევის შესახებ, შეგვიძლია დავასკვნათ, რომ ***დ. ბარონისაგან*** განსხვავებით ბევრს დანახვის შემდეგაც უჭირს ოპერაციული სისტემის ცნობა.

ოპერაციული სისტემის განსაზღვრების ფორმულირებაში შეიძლებოდა ჩამოგვეთვალა მის მიერ შესასრულებელი ფუნქციები, მაგრამ ვინაიდან ოპერაციულ სისტემებად წოდებულ კონკრეტულ პროგრამულ პროდუქტში შეიძლება ზოგიერთი მათგანი არ სრულდებოდეს, ამიტომ უმჯობესად მივიჩნიეთ **ოპერაციული სისტემა** ვუწოდოთ კომპიუტერის აპარატურის გამოყენების შესაძლებლობის უზრუნველმყოფი პროგრამების ერთობლიობას. ამსტერდამის თავისუფალი უნივერსიტეტის პროფესორის ***ე. ტანენბაუმის*** (***A. S. Tanenbaum, 1944***) მოხ-

დენილი გამოთქმის თანახმად ოპერაციული სისტემა წარმოადგენს პროგრამების კომპლექსს, რომელიც კომპიუტერის აპარატურულ სიმახინჯეს გარდაქმნის სილამაზედ და ამით კომპიუტერთან ურთიერთობისაკენ მონუსხულივით გვიზიდავს.

გამოყოფენ ოპერაციული სისტემის შემდეგ ძირითად ფუნქციებს:

- მომხმარებლის ინტერფეისის განსაზღვრა;
- მომხმარებლებს შორის აპარატურული რესურსების განაწილების უზრუნველყოფა;
- საერთო მონაცემებთან მუშაობის შესაძლებლობის უზრუნველყოფა;
- საერთო რესურსებთან მომხმარებელთა დაშვების დაგეგმვა;
- შეტანა/გამოტანის ოპერაციების ეფექტურად შესრულების უზრუნველყოფა;
- შეცდომების შემთხვევაში როგორც ინფორმაციის, ისე გამოთვლითი პროცესის აღდგენა.

ოპერაციული სისტემის განკარგულებაშია აქტიური და პასიური რესურსები. ***აქტიური რესურსი*** ეწოდება მმართველ, ხოლო ***პასიური რესურსი*** - მართვად რესურსებს. აქტიური რესურსებია: კომპიუტერის ოპერატორები, გამოყენებითი და სისტემური დამპროგრამებლები, ადმინისტრატიული პერსონალი, მომხმარებლის პროგრამები. პასიური რესურსებია: პროცესორები, მეხსიერება, შეტანა/გამოტანის მოწყობილობა, მონაცემები.

ოპერაციული სისტემა კომპიუტერის აქტიურ და პასიურ რესურსებს შორის ასრულებს შუამავალის ფუნქციას. მისი მუშაობა ჩამოჰგავს ადრეულ გამომთვლელ მანქანებთან მომუშავე ოპერატორების მუშაობას, რომლებიც შუამავლის ფუნქციებს ასრულებდნენ დავალებათა შემკვეთ კლიენტებსა და ელექტრონულ გამოთვლელ მანქანას შორის.

არსებობს ოპერაციული სისტემების კლასიფიცირების რამდენიმე ვარიანტი. ერთ-ერთი მათგანის ბლოკური სქემა ***1.1*** ნახაზზეა მოყვანილი. აღნიშნული სქემის თანახმად კლასიფიცირება მოხდენილია ოთხი ძირითადი ნიშნის, კერძოდ, რესურსების მართვის ალგორითმების თავისებურებებს, აპარატურული პლატფორმის თავისერებების, ოპერაციული სისტემის გამოყენების სფეროებისა და ოპერაციული

სისტემის აგების კონცეფციის მიხედვით. მოკლედ განვიხილოთ თითოეული მათგანი

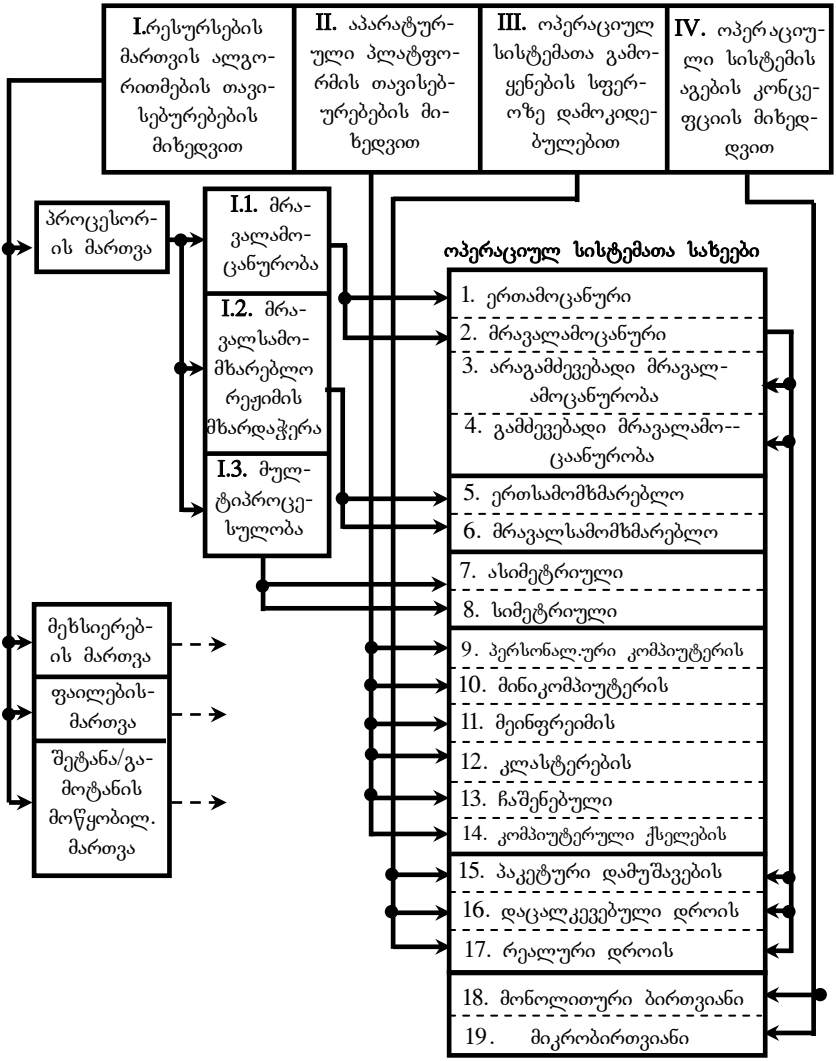
■ ***I.რესურსების მართვის ალგორითმების თავისებურებები.*** კომპიუტერის პასიური რესურსების მართვის ალგორითმების ეფექტურობაზე მნიშვნელოვანწილადაა დამოკიდებული მთლიანად ოპერაციული სისტემის ეფექტურობა. ამიტომ ოპერაციული სისტემის დახასიათებისას ხშირად მოჰყავთ პროცესორების, მეხსიერების, გარე მოწყობილობების სამართავად ოპერაციული სისტემის მიერ შესასრულებელი ფუნქციების რეალიზაციის თავისებურებები. შევნიშნავთ, რომ არსებული რესურსებიდან (პროცესორი, მეხსიერება, ფაილების სისტემა და შეტანა/გამოტანის მოწყობილობები) ***1.1.*** ნახაზზე კლასიფიცირება მოხდენილია მხოლოდ პროცესორის მართვის ალგორითმების თავისებურებების მიხედვით.

ოპერაციული სისტემის მიერ პროცესორის მართვის გამოყენებული ალგორითმის თავისებურებაზე დამოკიდებულებით ოპერაციული სისტემები იყოფა (იხ. ნახ.***1.1***) ერთამოცანურ და მრავალამოცანურ, ერთსამომხმარებლო და მრავალსამომხმარებლო სისტემებად. მოკლედ დავახასიათოთ თითოეული მათგანი.

■ ***ერთამოცანური ოპერაციული სისტემა*** ძირითადად მომხმარებლისათვის ვირტუალური მანქანის მიწოდების ფუნქციას ასრულებს, რითაც ადვილსა და მოსახერხებელს ხდის კომპიუტერთთან მომხმარებლის ურთიერთობის პროცესს; გარდა ამისა, აღნიშნული სისტემა მართავს პერიფერიულ მოწყობილობებს, ფაილებს და უზრუნველყოფს კომპიუტერთან მომხმარებლის ურთიერთობას.

■ ***მრავალამოცანური ოპერაციული სისტემა*** ასრულებს ზემოთ ჩამოთვლილ ყველა ფუნქციას და, გარდა ამისა, კომპიუტერში ერთდროულად მიმდინარე პროცესებს შორის ანაწილებს ისეთ რესურსებს, როგორიცაა პროცესორული დრო, ოპერატიული მეხსიერება, ფაილები და გარე მოწყობილობები.

კომპიუტერში მიმდინარე რამდენიმე პროცესს შორის პროცესორული დროის განაწილების გამოყენებული ხერხი მნიშვნელოვანწილად განსაზღვრავს ოპერაციული სისტემის სპეციფიკას. მრავალამოცანურობის რეალიზაციის არსებულ ვარიანტებიდან ***1.1*** ნახაზზე გამოყოფილია ორი ჯგუფის ალგორითმები, რომელთა დროსაც რეალიზდება:



ნახ.1.1. *ოპერაციულ სისტემათა კლასიფიცირების ბლოკური სქემა*

1)არაგამძევებადი მრავალამოცანური ოპერაციული სისტემა*(NetWare, Windows 3.x);*

2) გამძევებადი მრავალამოცანური ოპერაციული სისტემა*(Windows NT,OS/2, UNIX).*

ისინი ერთმანეთისაგან ძირითადად პროცესების დაგეგმვის ცენტრალიზაციის ხარისხით განსხვავდება. პირველ შემთხვევაში პროცესებს მთლიანად გეგმავს ოპერაციული სისტემა, ხოლო მეორე შემთხვევაში – ერთობლივად ოპერაციული სისტემა და გამოყენებითი პროგრამა. ***არაგამძევებადი მრავალამოცანურობის*** დროს აქტიური პროცესი სრულდება მანამ, სანამ იგი თავად არ გადასცემს მართვას ოპერაციულ სისტემას (მორიგი მომზადებული პროცესის ამორჩევის მიზნით). ***გამძევებადი მრავალამოცანურობის*** დროს ერთი პროცესიდან მეორეზე გადართვის გადაწყვეტილებას ერთპიროვნულად ოპერაციული სისტემა იღებს.

მრავალსამომხმარებლო რეჟიმის მხარდაჭერის მიხედვით არსებობს ორი სახის ოპერაციული სისტემა (იხ. ნახ. ***1.1***):

1) ერთსამომხმარებლო ოპერაციული სისტემა *(MS-DOS, Windows 3.x, OS/2-ის ადრეული ვერსიები);*

2) მრავალსამომხმარებლო ოპერაციული სისტემა *(UNIX, Windows NT).*

მრავასამომხმარებლო ოპერაციული სისტემა ერთსამომხმარებლო სისტემისაგან ძირითადად იმით განსხვავდება, რომ მასში თითოეული მომხმარებლის კუთვნილი ინფორმაცია დაცულია სხვა მომხმარებლის არასანქცირებული შეღწევისაგან. ხაზგასასმელია ის გარემოება, რომ ყველა მრავალამოცანური ოპერაციული სისტემა არ არის მრავალსამომხმარებლო და, ასევე, ყველა ერთსამომხმარებლო სისტემა არ არის ერთამოცანური.

■ ***მულტიპროცესორობა*** ნიშნავს ინფორმაციის დამუშავების პროცესში მრავალი (რამდენიმე) პროცესორის მონაწილეობას. იგი ართულებს რესურსების მართვის ალგორითმებს. მულტიპროცესორები გამოიყენება *IBM* ფირმის მიერ დამუშავებულ *OS/2*, *„Microsoft“* ფირმის მიერ დამუშავებულ *Windows NT*, *„Novell“* ფირმის მიერ დამუშავებულ*NetWare* და მრავალ სხვა ოპერაციულ სისტემაში.

მულტიპროცესოცესორული არქიტექტურიის მქონე სისტემაში გამოთვლითი პროცესის ორგანიზების ხერხზე დამოკიდებულებით განასხვავებენ (იხ. ***ნახ. 1.1***) ასიმეტრიულ და სიმეტრიულ ოპერარაციულ სისტემებს.

■ ***ასიმეტიული მულტიპროცესორული სისტემა*** ნაწილობრივადაა დეცენტრალიზებული: გამოყენებით ამოცანებს მასში ერთი პროცესორი ანაწილებს დანარჩენ პროცესორებს შორის; ***სიმეტრიული მულტიპროცესორული სისტემა*** მთლიანადაა დეცენტრალიზებული: იგი იყენებს პროცესორების მთელ ***პულს***, რომელსაც ანაწილებს სისტემური და გამოყენებით ამოცანებს შორის.

შევნიშნავთ, რომ ინგლისური სიტყვა **Pool** ანუ ***პული*** ნიშნავს „ფონდს“, „ჯგუფს“. ***საერთო პროცესორი*** ეწოდება ფიზიკურ პროცესორებს, რომელთა პროცესორული სიმძლავრე რამდენიმე ლოგიკურ განყოფილებების მიერ არის განაწილებული. რამდენიმე ლოგიკურ განყოფილებებს შორის ფიზიკურ პროცესორთა პროცესორული სიმძლავრის განაწილებას *Micro-Partitioning* ეწოდება. საერთო პულში შემავალი პროცესორები შეიძლება ნაწილ-ნაწილ მიითვისოს ლოგიკურმა განყოფილებებმა.

■ ***<u>II. აპარატურული პლატფორმების თავისებურებების</u>*** მიხედვითგანასხვავებენ (იხ. ნახ. 1.1): პერსონალური კომპიუტერებისათვის, მინი-კომპიუტერებისათვის, მეინფრემებისათვის, კლასტერებისათვის, ჩაშენებული კომპიუტერებისათვის და კომპიუტერული ქსელისათვის განკუთვნილ ოპერაციულ სისტემას. ზემოთ ჩამოთვლილ სისტემებს შეიძლება დავუმატოთ სერვერებისა და სმარტ-ბარათების ოპერაციული სისტემები.

■ ***<u>III. ოპერაციული სისტემის გამოყენების გარემოს თავისებურებებზე</u>***დამოკიდებულებით მრავალამოცანური ოპერაციული სისტემები მისდამი წაყენებულ ეფექტურობის კრიტერიუმზე დაყრდნობით იყოფა (იხ. ნახ. ***1.1***):

1) პაკეტური დამუშავების ოპერაციულ სისტემებად;
2) დროის დაცალკევების ოპერაციულ სისტემებად;
3) რეალური დროის ოპერაციულ სისტემებად.

■ ***პაკეტური დამუშავების ოპერაციული*** სისტემებიძირითადად გამოთვლითი ხასიათის ამოცანების გადასაწყვეტადაა განკუთვნილი. მისდამი შერჩეული***ეფექტურობის კრიტერიუმია*** მაქსიმალური გამტარობის უნარის უზრუნველყოფა, ე. ი. დროის ერთეულში მაქსიმალური რაოდენობის ამოცანების გადაწყვეტა. ამ კრიტერიუმის შესაბამისად პაკეტური დამუშავების სისტემებში გამოიყენება ფუნქციონირების შემდეგი სქემა: მუშაობის დასაწყისში ფორმირდება ***დავალებათა პაკეტი***, რომელშიც თითოეული ამოცანა შეიცავს სისტემური რესურსებისათვის წასაყენებელ მოთხოვნებს. დავალებათა ამ

პაკეტისაგან ფორმირდება ***მულტიპროგრამული ნარევი***, ე. ი. ერთდროულად შესასრულებელი ამოცანების სიმრავლე. ერთდროულად შესასრულებლად შეირჩევა ამოცანები, რომლებიც რესურსებს ისეთ განსხვავებულ მოთხოვნებს წაუყენებს, რომ კომპიუტერის ყველა მოწყობილობა ბალანსირებულად იქნეს დატვირთული; მაგალითად, მულტიპროგრამულ ნარევში სასურველია ერთდროულად შევიდეს გამოთვლითი ამოცანები და ინტენსიური შეტანა/გამოტანის მქონე ამოცანები. ამგვარად, დავალებათა პაკეტიდან თითოეული ახალი დავალება ამოირჩევა სისტემაში წარმოქმნილ სიტუაციაზე დამოკიდებულებით, ე. ი. ამოირჩევა „ხელსაყრელი" („მომგებიანი") დავალება. მაშასადამე, აღნიშნულ ოპერაციულ სისტემებში დროის გარკვეულ პერიოდში ამა თუ იმ პროგრამის შესრულება გარანტირებული არ არის. პაკეტური დამუშავების სისტემებში პროცესორი ერთი ამოცანის შესრულებიდან მეორეზე მხოლოდ მაშინ გადაირთვება, როდესაც აქტიური ამოცანა თავად იტყვის უარს პროცესორზე, მაგალითად იმისათვის, რომ შეასრულოს შეტანა/გამომოტანის ოპერაცია. ამიტომ ერთმა ამოცანამ შეიძლება ძალიან დიდი ხნით დაიკავოს პროცესორი, რის შედეგადაც შეუძლებელია ინტერაქტიული ამოცანების შესრულება. ამგვარად, კომპიუტერთან, რომელზეც დაყენებულია პაკეტური დამუშავების სისტემა, მომხმარებლის ურთიერთობა დაიყვანება იმაზე, რომ იგი დისპეტჩერ-ოპერატორს ჩააბარებს დავალებას და პასუხს მიიღებს მხოლოდ მას შემდეგ, როდესაც კომპიუტერი დაამთავრებს დავალებათა მთელი პაკეტის შესრულებას. ეს ამცირებს მომხმარებლის მუშაობის ეფექტურობას. მაშასადამე, ***პაკეტური დამუშავების ოპერაციული სისტემის ნაკლია*** ის, რომ მომხმარებელი-დამპროგრამებელი იზოლირებულია მისი ამოცანების დამუშავების პროცესისაგან.

■ ***დაცალკევებული დროის ოპერაციული სისტემები*** (მაგ., *UNIX, RT/11*) შეიქმნა პაკეტური დამუშავების ოპერაციული სისტემისათვის დამახასიათებელი ზემოთ აღნიშნული ნაკლის აღმოსაფხვრელად. ***მისი ეფექტურობის კრიტერიუმია*** არა მაქსიმალური გამტარობის უნარის, არამედ მომხმარებლის მუშაობის ეფექტურობის უზრუნველყოფა.

სისტემის ეფექტურობის დაცალკევებული დროის ოპერაციული სისტემის თითოეულ მომხმარებელს გამოეყოფა ინდივიდუალური ტე-

რმინალი, რომლიდანაც მას შეუძლია დიალოგი გამართოს საკუთარ პროგრამასთან. ასეთ სისტემაში თითოეულ ამოცანას ეთმობა პროცესორული დროის მხოლოდ კვანტი, ამიტომ იგი დიდი ხნის განმავლობაში ვერ იკავებს პროცესორს და პასუხის დრო მისაღები ხდება მომხმარებლისათვის. საკმაოდ მცირე სიდიდის კვანტის შერჩევის შემთხვევაში ერთსა და იმავე მანქანასთან მომუშავე ყველა მომხმარებელს ექმნება შთაბეჭდილება, რომ თითოეული მათგანი ერთპიროვნულად ფლობს მანქანას. ნათელია, რომ დაცალკევებული დროის ოპერაციულ სისტემას პაკეტურ სისტემაზე უფრო ნაკლები გამტარობის უნარი გაქვს, რადგან იგი ასრულებს არა იმ ამოცანას, რომელიც მისთვის „მომგებიანი", არამედ ერთდროულად ყველა მათგანს, და, გარდა ამისა, ერთი ამოცანიდან მეორეზე ხშირი გადართვები დამატებით ზრდის პროცესორის ზედნადებ ხარჯს. სამაგიეროდ იზრდება მომხმარებლის მუშაობის ვეფექტურობა, ე.ი. იგი აკმაყოფილებს ეფექტურობის მისთვის შერჩეულ კრიტერიუმს.

■ ***რეალური დროის ოპერაციული სისტემა*** (მაგალითად, *QNX, RT/11*). არსებობს ობიექტები, რომელთა მართვის პროგრამის დროულად შეუსრულებლობა შეიძლება ავარიის მიზეზი გახდეს. მაგალითად, დროულად თუ არ შესრულდა თანამგზავრის მართვის პროგრამა, თანამგზავრმა შეიძლება დატოვოს ხილვადობის ზონა; ასევე, საექსპერიმენტო დანადგარიდან მოსული მონაცემების ფიქსირების პროგრამის დროულად შეუსრულებლობისას შეიძლება დაიკარგოს აღნიშნული მონაცემები. არსებობს უამრავი მსგავსი ობიექტები. სწორედ მათთვისაა აუცილებელი გამოვიყენოთ რეალური დროის გარკვეული ოპერაციული სისტემა. აღნიშნულიდან გამომდინარე, ***რეალური დროის ოპერაციული სისტემის ეფექტურობის კრიტერიუმია,*** დაიცვას პროგრამების ამუშავებისა და შედეგის მიღების წინასწარ დასახული დროითი ინტერვალები.აღნიშნულ დროს ეწოდება ***სისტემის რეაქცია***, ხოლო სისტემის ზემოთ აღწერილ თვისებას – ***სისტემის რეაქციულება.*** ასეთი სისტემებისათვის ***მულტიპლექსურ ნარევს*** წარმოადგენსწინასწარ დამუშავებული პროგრამების ფიქსირებული ნაკრები, რომლიდანაც შესასრულებლად პროგრამა ამოირჩევა ობიექტის მიმდინარე მდგომარეობის ან წინასწარ დაგეგმილი სამუშაოთა განრიგის შესაბამისად.

ზოგიერთ ოპერაციულ სისტემებში შეთავსებულია სხვადასხვა ტიპის სისტემათა თვისებები. მაგალითად, ამოცანების ნაწილი შეიძლება პაკეტურად დამუშავდეს, ნაწილი კი – რეალური დროის რეჟიმში ან დაცალკევებული დროის რეჟიმში. ასეთ შემთხვევებში პაკეტური დამუშავების რეჟიმს ხშირად – ***ფონურ რეჟიმს***უწოდებენ.

<u>IV. აგების კონცეფციაზე დამოკიდებული ოპერაციული სისტემები</u>ოპერაციული სისტემის აღწერისას ხშირად მიუთითებენ მისი სტრუქტურული ორგანიზაციის თავისებურებებსა და მასში ჩადებულ ძირითად კონცეფციებს.

ასეთ საბაზისო კონცეფციებს მიეკუთვნება ***მონოლითიური ბირთვი*** ან ***მიკრობირთვული მიდგომა***. ოპერაციული სისტემების უმრავლესობაში გამოყენებულია ***მონოლითური ბირთვი***; იგი პრივილეგირებულ რეჟიმში მომუშავე ერთი პროგრამის სახით აიგება, რომელიც აღნიშნული რეჟიმიდან გამოუსვლელად სწრაფად გადადის ერთი პროცესიდან მეორეზე და პირიქით. ალტერნატიულ მიდგომას წარმოადგენს ოპერაციული სისტემის აგება აგრეთვე პრივილეგირებულ რეჟიმში მომუშავე ***მიკრობირთვის*** ბაზაზე; იგი ასრულებს აპარატურის მართვის მინიმალური რაოდენობის ფუნქციას, ხოლო ოპერაციული სისტემის უფრო მაღალი დონის ფუნქციებს ასრულებს ოპერაციული სისტემის სპეციალიზებული კომპონენტები – სამომხმარებლო რეჟიმში მომუშავე სერვერები. ამგვარად აგებული ოპერაციული სისტემა უფრო ***ნელა მუშაობს***, რადგან მას ხშირად უხდება პრივილეგირებულ და სამომხმარებლო რეჟიმებს შორის გადასვლები, სამაგიეროდ იგი ***უფრო მოქნილია***: სამომხმარებლო რეჟიმის სერვერების დამატების, მოდიფიცირების ან გამორიცხვის მეშვეობით შესაძლებელია სისტემის გაზრდა, მოდიფიცირება ან შევიწროება. გარდა ამისა, სერვერები ნებისმიერი სამომხმარებლო პროცესებივით ერთმანეთისაგან კარგადაა დაცული.

არსებობს ოპერაციული სისტემის აგების სხვა კონცეფციებიც, რომელთა შესაბამისადაც შეიძლება მოვახდინოთ აღნიშნული სისტემების კლასიფიცირება, მაგრამ ვიწრო სპეციფიკურობის გამო მათ არ განვიხილავთ.

დასასრულს აღვნიშნავთ, რომ ოპერაციულ სისტემა ***Windows NT****-ს განხილვისას მუდმივად გამოიყენება ცნებები „****სამომხმარებლო რეჟიმი****“და „****პრივილეგირებული (ანუ ბირთვის) რეჟიმი****“.*

***სამომხმარებლო რეჟიმი Windows NT**-ს მხარდამჭერი ყველაზე ნაკლებად პრივილეგირებული რეჟიმია; მას არ აქვს მოწყობილობებთან პირდაპირ შეღწევის უფლება, მეხსიერებასთან კი შეუძლია შეზღუდულად შეაღწიოს. **ბირთვის რეჟიმი – პრივილეგირებული რეჟიმია.** ამ რეჟიმში გამოყენებულ **NT**–ს ნაწილებს (მოწყობილობათა დრაივერებსა და „ვირტუალური მეხსიერების დისპეტჩერად“ წოდებულ ქვესისტემებს) აქვს ყველა მოწყობილობასთან და მეხსიერებასთან პირდაპირ შეღწევის უფლება. სამომხმარებლო და პრივილეგირებულ (ბირთვის) რეჟიმებს შორის განსხვავება უზრუნველყოფილია აპარატურულად (პროცესორის მიერ). (**NT** ანუ **New Technology** ნიშნავს ახალ **ტექნოლოგიას; Windows**-ის ყველა თანამედროვე ოპერაციული სისტემა ამ ტექნოლოგიაზეა დაფუძნებული)*

1.3. პროცესებისადანაკადებისცნებები

კომპიუტერული პროგრამა პროცესორის მიერ შესასრულებელი ინსტრუქციების პასიური მიმდევრობაა. მისი შესრულების აღმწერ აბსტრაქციას ***პროცესი*** ანუ ***ამოცანა*** ეწოდება. კომპიუტერში ერთდროულად მრავალი პროცესი სრულდება, ცალკე აღებული პროცესი კი ოპერაციული სისტემის სამუშაოს ერთეული, სისტემური რესურსების მოხმარებაზე ფორმულირებული განაცხადია.

პროცესორის მიერ გარკვეულ პროგრამის შესრულებაზე დახარჯულ დროს ***პროცესორული დრო*** (ინგლ. ***processtime***, ანუ ***CPUtime***) ეწოდება. ოპერაციული სისტემა შეიცავს ***პროცესების მართვის ქვესისტემას,*** რომლის დანიშნულებაა: ***1)*** შექმნას და გააუქმოს პროცესი; ***2)*** პროცესორული დრო გადაანაწილოს სისტემაში ერთდროულად მიმდინარე პროცესებს შორის; ***3)*** დაგეგმოს პროცესის შესრულება; ***4)*** პროცესები უზრუნველყოს აუცილებელი სისტემური რესურსებით; ***5)*** უზრუნველყოს პროცესების ურთიერთზემოქმედება.

პროცესს ზოგიერთი რესურსი თავიდანვე გამოეყოფა, ნაწილი კი მას შესრულების პროცესში მიეწოდება ამის შესახებ ფორმირებული მოთხოვნების კვალობაზე; რესურსების ნაწილი პროცესის განკარგულების ქვეშ რჩება ამ პროცესის დამთავრებამდე, ნაწილს კი იგი დროის მხოლოდ ფიქსირებული პერიოდის განმავლობაში ფლობს. ***პროცესების მართვის ქვესისტემა*** საკუთარი ფუნქციების შესასრულებლად ურთიერობს რესურსების მართვის სხვა ისეთ ქვესისტემებთან, როგორიცაა მეხსიერების მართვის ქვესისტემა, შეტანა/გამო-

ტანის ქვესისტემა, ფაილურ სისტემა და ა. შ. დამატებითი პრობლემები წარმოიშობა ერთდროულად რამდენიმე დამოუკიდებელი პროცესის შესრულებისას. მიუხედავად იმისა, რომ პროცესები წარმოიქმნება და სრულდება ასინქრონულად, შეიძლება მათ შორის მაინც მოხდეს ურთიერთზემოქმედება, მაგალითად, მონაცემების გაცვლის დროს. გარდა ამისა, ხშირად რამდენიმე პროცესი ცდილობს შეცვალოს ერთი და იგივე ფაილი; ამ დროს შეიძლება წარმოიშვას ე. წ. ***„შეჯიბრის“ ეფექტი***, რომლის დროსაც თითოეული პროცესი ცდილობს „დაასწროს“ მეორე პროცესს; აღნიშნული ეფექტის, აგრეთვე ურთიერთბლოკირებებისა და სხვა კოლიზიის გამოსარიცხად, ძალიან მნიშვნელოვანი ხდება პროცესების მიმდინარეობის სიჩქარეთა ურთიერთშეხამება, ანუ ***პროცესების სინქრონიზება***პროცესების მართვის ქვესისტემის ერთ-ერთი უბნიშვნელოვანესი ფუნქციაა.

პროცესის დასრულებისას პროცესების მართვის ქვესისტემა ხურავს პროცესის მიერ გამოყენებულ ყველა ფაილს; ათავისუფლებს ოპერატიული მეხსიერების მისთვის გამოყოფილ უბნებს (ახალი კოდების, მონაცემებისა და სისტემური საინფორმაციო სტრუქტურების განსათავსებლად); კორექტირებას უკეთებს ოპერაციული სისტემის მომსახურების მომლოდინე პროცესების რიგსა და იმ სიებს, რომლებშიც დამოწმებული იყო უკვე დასრულებული პროცესი.

■ ***მულტიდაპროგრამების მხარდასაჭერად*** ოპერაციულმა სისტემამ თავისთვის უნდა განსაზღვროს სამუშაოს ის შინაგანი ერთეულები, რომელთა შორის ნაწილდება პროცესორი და კომპიუტერის სხვა რესურსები. უნდა გვახსოვდეს, რომ ***მულტიდაპროგრამება*** გამოთვლითი პროცესის იმგვარად ორგანიზების ხერხია, რომლის დროსაც კომპიუტერის მეხსიერებაში არსებობს პროცესორზე მონაცვლეობით შესასრულებელი რამდენიმე პროგრამა.

დღეისათვის ოპერაციული სისტემების უმრავლესობაში განსაზღვრულია სამუშაოს ***ორი ტიპის ერთეული***. სამუშაოს უფრო მსხვილ ერთეულს წარმოადგენს მოცემული პარაგრაფის დასაწყისში აღშნული ***პროცესი*** ანუ ***ამოცანა***, რომლის შესასრულებლად აუცილებელია ***„ნაკადად“*** ანუ ***„ძაფად“*** წოდებული რამდენიმე უფრო წვრილი სამუშაოს შესრულება.

ნებისმიერი გამოთვლითი სისტემა გარკვეულ პროგრამებს ასრულებს, ამიტომ როგორც პროცესს, ისე ნაკადს უკავშირდება გარკვეული პროგრამული კოდი, რომელიც ამისათვის იღებს ***შესასრულებელი მოდულის*** სახეს. პროგრამული კოდის შესასრულებლად აუცილებელია იგი ჩაიტვირთოს ოპერატიულ მეხსიერებაში; მონაცემების შესანახად შესაძლებელია მას დისკზე გამოეყოს გარკვეული ადგილი, მიეცეს შეტანა/გამოტანის მოწყობილობებთან შეღწევის უფლება, მაგალითად, მიმდევრობითი პორტით. შესრულების პროცესში პროგრამას შეიძლება დასჭირდეს შეღწევა საინფორმაციო რესურსებთან, მაგალითად, ფაილებთან და, რა თქმა უნდა, შესაძლებელია პროგრამა შესრულდეს მისთვის პროცესორული დროის მიუცემლად, რომლის განმავლობაში პროცესორი ასრულებს მოცემული პროგრამის კოდებს.

განვიხილოთ გამოთვლითი სისტემა, რომელიც შეიცავს როგორც პროცესებს, ასევე ნაკადებსაც; ოპერაციული სისტემა პროცესს გამოუყოფს ნებისმიერი სახის საჭირო რესურს, გარდა ***პროცესორული დროისა***. ამ უკანასკნელს იგი ანაწილებს ნაკადებს შორის. ***ნაკადი*** პროცესის შემადგენელი ნაწილია და შედგება მიმდევრობითად (ერთმანეთის მიყოლებით) შესასრულებელი ბრძანებებისაგან.

1980 წლამდე მიიჩნევდნენ, რომ პროცესი მხოლოდ ერთი ნაკადისაგან შედგებოდა. ასეთი შეხედულება შენარჩუნებულია ზოგიერთ თანამედროვე ოპერაციულ სისტემებშიც. მათში პროცესის ცნება შთანთქავს ნაკადის ცნებას და ამიტომ ასეთ სისტემებში ***მულტიდაპროგრამება*** პროცესების დონეზე ხდება. სინამდვილეში ერთი ნაკადის შემცველი პროცესი მხოლოდ უმარტივესი სახის პროცესია.

პროცესი შეიძლება იყოს ***„მზა“, „შესრულებად“*** ან ***„ბლოკირებულ“*** მდგომარეობაში.

ერთპროცესორულ სისტემაში დროის ნებისმიერ კონკრეტულ მომენტში შეიძლება სრულდებოდეს ერთადერთი პროგრამა, დანარჩენი მზა პროგრამები ბლოკირებულ მდგომარეობაშია, მათ მინიჭებული აქვს პრიორიტეტები, რომელთა დაცვით დგას შესასრულების რიგში.

პროცესები ერთდროულად რომ ვერ ჩაერიოს რესურსების განაწილებაში, აგრეთვე მათ რომ არ დააზიანის ერთმანეთის კოდები და მონაცემები, ოპერაციული სისტემა პროცესებს ერთმანეთისაგან განამხოლოებს. ამისათვის ოპერაციული სისტემა თითოეულ პროცესს

უზრუნველყოფს ვირტუალური სამისამართო სივრცით, რის მეოხებითაც შეუძლებელია რომელიმე პროცესმა შეაღწიოს მეორე პროცესის ბრძანებებთან და მონაცემებთან.

■ ***პროცესის ვირტუალური სამისამართო სივრცე*** წარმოადგენს იმ მისამართების ერთობლიობას, რომლებითაც მანიპულირება შეუძლია მხოლოდ პროგრამულ მოდულს.

ოპერაციული სისტემა ვირტუალურ სამისამართო სივრცეს ასახავს ფიზიკური მეხსიერების გამოყოფილ პროცესზე.

ერთმანეთს შორის ურთიერთქმედების აუცილებლობის შემთხვევაში პროცესები მიმართავს ოპერაციულ სისტემას, რომელიც შეასრულებს შუამავალის ფუნქციებს და ერთმანეთთან კავშირის დასამყარებლად მათ გამოუყოფს კონვეიერებს, საფოსტო ყუთებს, მეხსიერების ცალკეულ (დაყოფილ) სექციებს.

■ განვიხილოთ ოპერაციული სისტემები, რომლებშიც პროცესის ცნების მიერ მთლიანადაა შთანთქმული ნაკადის ცნება და სამუშაოს ერთეულს მარტო პროცესი წარმოადგენს.

ასეთი სისტემაში პროგრამების რაოდენობის უბრალო გაზრდა ვერ გაიზრდის გამტარობის უნარს, ვინაიდან ცალკე აღებული პროცესი ვერ შესრულდება იმაზე უფრო სწრაფად, ვიდრე იგი სრულდება ერთპროგრამულ რეჟიმში; რესურსების ნებისმიერი გადანაწილება მხოლოდ შეანელებს ერთ-ერთი პროცესის სიჩქარეს, რადგან მას მოუხდება საჭირო რესურსის ლოდინში დაკარგოს დრო.

სისტემის გამტარობის ასამაღლებლად საჭირო იქნება გამოვიყენოთ პროცესების დაპარელელების ტრადიციული ხერხები. კერძოდ:

1) დაპარალელების რთული ამოცანის გადაწყვეტა თავის თავზე უნდა აიღოს გამოყენებითი სფეროს ***დამპროგრამებელმა*** და მის მიერ შედგენილ პროგრამებში გაითვალისწინოს გამოთვლების ამა თუ იმ შტოებისათვის მართვის პარალელურად გადამცემი სპეციალური ***პროგრამა-დისპეტჩერი***. ამ დროს მიიღება მართვის მრავალრიცხოვანი გადაცემების შემცველი ლოგიკურად მეტად აბურდული პროგრამა, რომლის ***გამართვა და მოდიფიცირება ძალზე რთული იქნება***;

2) ერთი პროგრამის თითოეული პარალელური შტოსათვის შევქმნათ საკუთარი პროცესი, ე.ი. ერთი პროგრამა წარმოვადგინოთ თითო პროცესისაგან შემდგარ რამდენიმე პროგრამის სახით. ეს პროგრამები ერთ ამოცანას წყვეტს, ამიტომ მათ ბევრი რამ შეიძლე-

ბა ჰქონდეს საერთო, კერძოდ, შეიძლება ისინი იყენებდეს ერთსა და იმავე მონაცემებს და/ან მეხსიერების სეგმენტებს; ჰქონდეს რესურსებში შეღწევის ერთნაირი უფლებები და ა. შ. აღნიშნულის გამო ამ პროგრამების რეალიზებისათვის როგორც ოპერაციულ სისტემებს, ისე პროცესებს უნდა ჰქონდეს გამოთვლების დაპარალელების ისეთი მექანიზმი, რომელშიც გათვალისწინებული იქნება ერთსა და იმავე პროგრამის გამოთვლათა ცალკეულ შტოებს შორის არსებული მჭიდრო კავშირები. ძველ ოპერაციულ სისტემებს ასეთი მექანიზმები არ აქვს.

ზემოთ აღნიშნული ნაკლის დასაძლევად თანამედროვე ოპერაციული სისტემებში შემტანილია ***მრავალნაკადურიდამუშავების მექანიზმი*** (***multithreading***). კერძოდ, ფორმირდება სამუშაოს ახალი ერთეული – ***შესრულების ნაკადი*** რაც მნიშვნელოვნად ცვლის პროცესის შინაარსს. „***ნაკადის***“ ცნებაში იგულისხმება პროგრამის იმ ბრძანებათა ერთობლიობა, რომლის ***ერთი წევრიდან მეორეზე პროცესორი გადადის მიმდევრობით***. ოპერაციული სისტემა პროცესორულ დროს ***ნაკადებს შორის ანაწილებს***, ხოლო თითოეულ პროცესს უნიშნავს სამისამართო სივრცესა და რესურსების ნაკრებს, რომლებიც უნდა გამოიყენოს ამ პროცესის ყველა ნაკადმა.

ხაზს ვუსვამთ იმ ფაქტს, რომ ერთპროგამულ სისტემებში საჭირო არ არის ნაკადის ცნების შემოტანა, რადგან მათში არ არსებობს რესურსების დანაწილების პრობლემა.

■ ნაკადების შექმნა ოპერაციულ სისტემას უფრო ნაკლებ ზედნადებ ხარჯებს თხოვს, ვიდრე პროცესები. კონკრეტული პროგრამა (გამოყენება) ერთადერთ პროცესს წარმოქმნის, რომელიც რამდენიმე ნაკადს უდევს დასაბამს. ამიტომ ***ოპერაციული სისტემა*** ნაკადებს ერთმანეთისაგან იმაზე გაცილებით ნაკლები სიმკაცრით განამხოლოებს, ვიდრე პროცესები ტრადიციულ მულტიპროგრამულ სისტემაში იყო ურთიერთგანმხოლოებული. კერძოდ, ერთი პროცესის ყველა ნაკადი იყენებს საერთო ფაილებს, ტაიმერებს, მოწყობილობებს, ოპერატიული მეხსიერების ერთსა და იმავე სფეროს, ერთსა და იმავე სამისამართო სფეროს. ეს ნიშნავს, რომ ისინი ერთმანეთს შორის ერთსა და იმავე გლობალურ ცვლადებს ინაწილებს. ვინაიდან თითოეული ნაკადი პროცესის ნებისმიერ ვირტუალურ მისმართთან დაიშვება, ამიტომ ერთ ნაკადს შეუძლია გამოიყენოს მეორე ნაკადის

სტეკი. ერთი პროცესის ნაკადები ერთმანეთისაგან არ არის სრულად დაცული: ეს შეუძლებელიცაა და საჭიროც არ არის. ერთმანეთთან ურთიერთობისათვის და მონაცემების გასაცვლელად ნაკადებს სრულებით არ სჭირდება ოპერაციულ სისტემა; ამისათვის საკმარისია მათ გამოიყენოს საერთო მეხსიერება: ერთი ნაკადი ჩაწერს მონაცემებს, მეორე კი წაიკითხავს მას. რაც შეეხება სხვადასხვა პროცესის ნაკადებს, ისინი ერთმანეთისაგან ძველებურად მკაცრადაა განმხოლოებული.

■ ***მულტიდაპროგრამება*** უფრო ეფექტურია ნაკადების და არა პროცესების დინეზე. თითოეულ ნაკადს აქვს ბრძანებათა საკუთარი მთვლელი და სტეკი. ერთი პროგრამის რამდენიმე ნაკადად ფორმირება აჩქარებს ამ პროცესის შესრულებას, რადგან მასში შემავალი ნაკადები შეიძლება ფსევდოპარალელურად (მულტიპროცესორულ სისტემებში კი – პარალელურად) შესრულდეს. მრავალნაკადურობა განსაკუთრებით ეფექტურად შეიძლება გამოვიყენოთ განაწილებული პროგრამების (გამოყენებების) შესასრულელად; მაგალითად, სერვერმა შეიძლება პარალელურად შეასრულოს რამდენიმე კლიენტისაგან ერთდროულად მოსული მოთხოვნა.

■ ნაკადების ფორმირება არა მარტო ამაღლებს სისტემის მწარმოებლურობას, არამედ უზრუნველყოფს ადვილად წასაკითხი პროგრამების შექმნას. განვიხილოთ ორი მაგალითი.

1) „ჩამწერ-წამკითხველ“ამოცანებში ერთი ნაკადი ჩაწერს ბუფერში, ხოლო მეორე – წაიკითხავს ამ ჩანაწერს. ცალკეულ პროცესებად მათი ფორმირება არავითარ მოგებას არ მოგვცემს, რადგან მათ მაინც მიმდევრობით უნდა გამოიყენოს ბუფერი.

2)კლავიატურიდან (***Del*** ან ***Break***) ***შეწყვეტები*** სასურველია ორი ნაკადის სახით წარმოვიდგინოთ, რომლიდანაც ერთ-ერთი ნაკადის ფუნქცია იქნება შეწყვეტის სიგნალის ლოდინი. ეს სამომხმარებლო დონეზე შეამცირებს საჭირო შეწყვეტების რაოდენობას.

■ მრავალი ნაკადის არსებობა გაცილებით ეფექტურია ***მულტიპროცესორულ სისტემებში***, რადგან ეს ნაკადები შეიძლება სხვადასხვა პროცესორზე პარალელურად (და არა ერთ პროცესორზე ფსევდოპარალელურად) შესრულდეს.

■ პროცესის შექმნასთან ერთად აუცილებელია შევქმნათ ამ პროცესის აღმწერი; ***პროცესის აღმწერი*** ეწოდება საინფორმაციო სისტე-

მას (ან სისტემების ერთობლიობას), რომელშიც შენახული იქნება ყველა ის მონაცემი, რომელიც ოპერაციულ სისტემას დასჭირდება ამ პროცესის მართვისათვის. ასეთი მონაცემებია, მაგალითად, პროცესორის იდენტიფიკატორი, ინფორმაცია მეხსიერებაში შემსრულებელი მოდულის განთავსების შესახებ, პროცესის პრივილეგერების ხარისხი (შეღწევის პრიორიტეტი და უფლება) და ა.შ.

პროცესის აღმწერებია ა) ***OS/360***-ში – ამოცანების მართვის ბლოკი (***Task Control Bloc – TCB***); ბ) ***OS/2***-ში – პროცესის მმართველი ბლოკი (***Process Control Bloc -TCB***); გ) ***UNIX***-ში – პროცესის დესკრიპტორი, გ) ***Windows NT***-ში ობიექტ-პროცესი (***object-process***);

პროცესის აღმწერის შექმნა სისტემაში გამოთვლით რესურსებზე კიდევ ერთი პრეტენდენტის გამოჩენას მოასწავებს. ამ მომენტიდან დაწყებული ოპერაციული სისტემის რესურსების განაწილებისას საჭიროა ახალი პროცესის მოთხოვნილებებიც იქნეს გათვალისწინებული.

პროცესის შექმნისას მისი შესასრულებელი პროგრამის კოდები და მონაცემები დისკიდან ოპერატიულ მეხსიერებაში უნდა გადაიტვირთოს. ამისათვის ოპერაციულმა სისტემამ დისკზე უნდა აღმოაჩინოს ასეთი პროგრამის ადგილსამყოფელი, გადაანაწილოს ოპერატიული მეხსიერება და ახალი პროცესის შემსრულებელ პროგრამას ამ მეხსიერებაში გამოიყოს გარკვეული ადგილი. შემდეგ საჭიროა მეხსიერებაში გამოყოფილ უბნებში წაიკითხული იქნეს პროგრამა და საჭიროებისამებრ შეიცვალოს მისი ცალკეული პარამეტრები.

ვირტუალური მეხსიერებიან სისტემებში საწყის მომენტში შეიძლება პროცესის კოდებისა და მონაცემების მხოლოდ ნაწილი ჩაიტვირთოს, რათა აუცილებლობის კვალობაზე მოხდეს დანარჩენი ნაწილის „გადატუმბვა“. არსებობს სისტემები, რომლებისთვისაც პროცესის შექმნის ეტაპზე ოპერატიულ მეხსიერებაში კოდებისა და მონაცემების ჩატვირთვა აუცილებელი არ არის; ამის ნაცვლად ***შესასრულებელი მოდული*** ფაილური სისტემის იმ კატალოგიდან, რომელშიც იგი თავდაპირველად იმყოფებოდა, გადაკოპირდება გადატუმბვის ადგილზე – კოდებისა და მონაცემების შესანახად დისკზე გამოყოფილ სპეციალურ არეში. ყველა ამ მოქმედების შესრულებისას ***პროცესებ-***

ის მართვის ქვესისტემა მჭიდროდ ურთიერთობს ***მეხსიერების მართვის ქვესისტემასთან*** და ***ფაილურ სისტემასთან***.

■***მრავალნაკადურ სისტემაში*** პროცესის შექმნის დროს ოპერაციული სისტემა თითოეული პროცესისათვის შესრულების სულ მცირე ***ერთ ნაკადს*** მაინც ქმნის. პროცესის შექმნის ანალოგურად ნაკადის შექმნის დროსაც ოპერაციული სისტემა უზრუნველყოფს სპეციალურ საინფორმაციო სტრუქტურას – ***ნაკადის აღმწერს***, რომელიც შეიცავს ნაკადის იდენტიფიკატორს, მონაცემებს შეღწევის უფლებებისა და პრიორიტეტების შესახებ, ნაკადის მდგომარეობას და სხვა ინფორმაციას. საწყის მდგომარეობაში ნაკადი (ან პროცესი, თუ საუბარია სისტემაზე, რომელშიც ცნება „ნაკადი“ არ არის განსაზღვრული), ***შეჩერებულ მდგომარეობაშია***. შესასრულებლად ნაკადის ამორჩევის მომენტი მოცემულ სისტემაში ***პროცესორული დროის*** მიწოდების წესისა და მოცემულ მომენტში არსებული ყველა ნაკადისა და პროცესის გათვალისწინებით ამოირჩევა. კოდები და მონაცემები თუ გადატუმბვის არეში იმყოფება, მაშინ პროცესის ნაკადის გააქტიურებისათვის აუცილებელია ოპერატიულ მეხსიერებაში არსებობდეს შესასრულებელი მოდულის ჩასატვირთი ადგილი.

■ მრავალ სისტემაში ნაკადმი შეიძლება მიმართოს ოპერაციულ სისტემას ე. წ. ***შთამომავლებისათვის - შვილებისათვის ნაკადების შექმნის თხოვნით***. ამდენად არსებობს მშობელი-ნაკადი და მისი შვილი-ნაკადებიც. სხვადასხვა ოპერაციულ სისტემაში სხვადასხვაგვარია ურთიერთობა ***მშობელ-ნაკადსა*** და ***შვილ-ნაკადს*** შორის. ზოგიერთ ოპერაციულ სისტემაში მშობელი და შვილი ნაკადები ***სინქრონულად მუშაობს***, რაც იმნას ნიშნავს, რომ მშობელი ნაკადის დამთავრების შემდეგ ოპერაციული წყვეტს ყველა მისი ***შვილ-ნაკადების*** შესრულებას. არსებობს ისეთი ოპერაციული სისტემებიც, რომლებშიც მშობელი და მისი შვილი ნაკადები ***ასინქრონულად მუშაობს***: შვილები, როგორც წესი, მემკვიდრეობით იღებს მშობლის მრავალ თვისებას და მშობლის დამთავრების შემდეგ განაგრძობს ფუნქციონირებას. მრავალ სისტემაში შვილ-ნაკადების გაჩენა წარმოადგენს პროცესებისა და ნაკადების შექმნის ძირითად მექანიზმს.

■ ***პროცესების დაგეგმვისას:*** ***1)***განისაზღვრება შესასრულებელი პროცესის შეცვლის დრო; ***2)*** მზა პროცესების რიგიდან შესასრუ-

ლებლად ამოირჩევა მორიგი პროცესი; ***3)*** კონტექსტები „ძველი“ პროცესიდან გადაირთება „ახალ“პროცესზე.

პირველი ორი ამოცანა სრულდება პროგრამული საშუალებებით, ხოლო მესამე ამოცანა უმეტეს შემთხვევაში – აპარატურულად.

არსებობს პროცესების დაგეგმვის მრავალი სხვადასხვა ალგორითმი. ისინი ზემოთ ჩამოთვლილ ამოცანებს სხვადასხვაგვარად წყვეტს და უზრუნველყოფს სხვადასხვა თვისების მულტიდაპროგრამებას. მრავალრიცხოვან ასეთ ალგორითმებს შორის ყველაზე უფრო ხშირად გვხვდება ორი ჯგუფის ალგორითმი, რომელთაგანაც პირველ ჯგუფში გაერთიანებულია ***დაკვანტვაზე დაფუძნებული ალგორითმები***, ხოლო მეორე მათგანში – ***პრიორიტეტებზე დაფუძნებული ალგორითმები***.

■ ***დაკვანტაზე დაფუძნებული ალგორითმების შესაბამისად*** აქტიური პროცესი მაშინ იცვლება, როდესაც: ***1)***პროცესი მთავრდება და ტოვებს სისტემას; ***2***) ხდება შეცდომა; ***3)*** პროცესი გადადის ლოდინის რეჟიმში; ***4)*** ამოიწურება მოცემული პროცესისათვის გამოყოფილი ***პროცესორული დროის კვანტი***.

პროცესი, რომელმაც ამოწურა საკუთარი კვანტი, გადადის მზა რეჟიმში და ელოდება მისთვის პროცესორული დროის ახალი კვანტის გამოყოფას; მისი ლოდინის პერიოდში გარკვეული წესის შესაბამისად მზა პროცესების რიგიდან ამოირჩევა ახალი პროცესი. ამგვარად, ვერცერთი პროცესი ხანგრძლივად ვერ დაიკავებს პროცესორს, ამიტომ ***დაკვანტვა ფართოდ გამოიყენება დროის დანაწილების სისტემებში.***

სხვადასხვა პროცესის შეიძლება გამოეყოს ერთნაირი ან განსხვავებული კვანტები. თითოეული პროცესისათვის შეიძლება გამოიყოს ფიქსირებული სიდიდის კვანტები ან ამ კვანტების სიდიდეები შეიძლება იცვლებოდეს პროცესის არსებობის სხვადასხვა პერიოდებში. პროცესებმა, რომლებმაც მთლიანად ვერ გამოიყენეს მათთვის გამოყოფილი კვანტები (მაგალითად, შეტანა/გამოტანის ოპერაციების შესასრულებლად გასვლის გამო), შეიძლება შემდგომი მომსახურებისათვის მიიღოს პრივილეგიის სახის კომპენსაცია. მზა პროცესების რიგი შეიძლება ციკლურად ორგანიზდეს შემდეგი წესის შესაბამისად: ***1)*** „პირველი შემოვიდა – პირველად მოხდა მისი მომსახურება“

(***FIFO*** – «***First In, First Out***»), ***2)***„ბოლო შემოვიდა – პირველად მოხდა მისი მომსახურება“ (***LIFO*** - «***LastIn, First Out***»);

■ პრიორიტეტებზე დაფუძნებულ ალგორითმებში გამოიყენება პროცესის პრიორიტეტის ცნება.

პრიორიტეტი წარმოადგენს გამომთვლელი მანქანის რესურსების, კერძოდ, პროცესორული დროის, გამოყენებისას პროცესის პრილეგიის განმსაზღვრელ რიცხვს: რაც უფრო მაღალია ალგორითმის პრიორიტეტი, მით უფრო მეტი პრივილეგიებით სარგებლობს იგი. პრიორიტეტის აღსანიშნავად გამოიყენება მთელი, წილადური, დადებითი ან უარყოფითი რიცხვები. რაც უფრო მაღალია პროცესის პრივილეგია, მით უფრო ნაკლებ ხანს უხდება მას რიგში დგომა. შესასრულებელი სამუშაოს მნიშვნელობაზე, ან ღირებულებაზე დამოკიდებულებით სისტემის პრიორიტეტს განსაზღვრავს სისტემის ადმინისტრატორი ან მას გარკვეული წესების საფუძველზე გამოითვლის ოპერაციული სისტემა. მინიჭებული პრიორიტეტი პროცესს შეიძლება შეუნარჩუნდეს მთელი მისი არსებობის განმავლობაში, ან გარკვეული კანონით შეიცვალოს იგი. ბოლო შემთხვევაში საქმე გვაქვს ***დინამიკურ პრიორიტეტთან***.

■ ერთმანეთისაგან განასხვავებენ ***ფარდობითი და აბსოლუტური პრიორიტეტებიან ალგორითმებს***. ორივე შემთხვევაში მზა პროცესების რიგიდან შესასრულებლად კონკრეტული პროცესი ერთნაირად ამოირჩევა: პროცესების გააქტიურება და, საჭიროებისამებრ მათი შეცვლა, პრიორიტეტების დაცვით ხდება; ოღონდ გააქტიურების პროცესი მიმართულია ზემოდან ქვემოთ, ხოლო შეცვლის პროცესი – ქვემოდან ზემოთ, ე. ი. რაც უფრო მაღალია პროცესის პრიორიტეტი, მით უფრო ადრე აქტიურდება, და რაც უფრო დაბალია პრიორიტეტი – მით უფრო ადრე შეიცვლება პროცესი. ***ფარდობით პრიორიტეტებიან სისტემაში*** აქტიური პროცესი მაშინ შეიცვლება, როდესაც თვითონ დატოვებს პროცესორს და გადავა „ბლოკირებულ“ მდგომარეობაში, ან მოხდება შეცდომა, ან დასრულდება პროცესი. ***აბსოლუტურ პრიორიტეტებიან სისტემაში*** აქტიური პროცესი მაშინაც შეიცვლება, როდესაც მზა პროცესების რიგში გაჩნდება უფრო მაღალი პრიორიტეტიანი პროცესი. ამ შემთხვევაში შეწყვეტილი პროცესი გადავა მზა პროცესების რიგში.

მრავალ ოპერაციულ სისტემაში პროცესების დაგეგმვისას გამოიყენება როგორც დაკვანტვის, ასევე პრიორიტეტების მეთოდი. მაგალითად, დაგეგმვა შეიძლება დაკვანტვაზე იყოს დაფუძნებული, ხოლო რიგიდან პროცესი ამოირჩევა და/ან კვანტის სიდიდის განსაზღვრა ხდებოდეს პრიორიტეტების მეთოდით.

■ ***პროცესი შეიძლება დაიგეგმოს*** არაგამძევებადი (***non-preemptive***) ან გამძევებადი (***preemptive***) პროცედურის შესასრულებლად, რის შედეგადაც შესაბამისად მიიღება არაგამოძევებადი ან გამძევებადი მრავალამოცანურობა.

არაგამძევებადი მრავალამოცანურობა (non-preemptive multitasking) პროცესების დაგეგმვის ისეთი ხერხია, რომლის დროსაც აქტიური პროცესი მანმადე სრულდება, სანამ იგი საკუთარი ინიციატივით არ გადასცემს მართვას ***ოპერაციულ სისტემის დამგეგმავს***, რათა ამ უკანასკნელმა მზა პროცესების რიგიდან შესასრულებლად ამოირჩიოს მორიგი პროცესი.

გამძევებადი მრავალამოცანურობა (preemptive multitasking)პროცესების დაგეგმვის ხერხია, რომლის დროსაც ერთი პროცესიდან მეორე პროცესზე პროცესორის გადართვის გადაწყვეტილებას ერთპიროვნულად იღებს ***ოპერაციული სისტემის დამგეგმავი***.

მრავალამოცანურობის არაგამძევებად და გამძევებად ვარიანტები ერთმანეთისაგან ძირითადად ამოცანების დაგეგმვის მექანიზმის ცენტრალიზაციის ხარისხით განსხვავდება. ***გამძევებადი მრავალამოცანურობის დროს***ამოცანების დაგეგმვის მექანიზმი მთლიანად ოპერაციულ სისტემაშია თავმოყრილი და დამპროგრამებელს არ აწუხებს ის, თუ როგორ შესრულდება მისი პროგრამა სხვა პროგრამების პარალეურად. ამ დროს ოპერაციული სისტემა: განსაზღვრავს აქტიური ამოცანის შესრულების შეწყვეტის მომენტს, იმახსოვრებს მის კონტექსტს, მზა პროცესების რიგიდან ამოირჩევს მორიგ პროცესს, ჩატვირთავს მის კონტექსტს და დაიწყებს ამ პროცესის შესრულებას. შესრულებას. ***არაგამძევებადი მრავალამოცანურობის დროს*** ამოცანების დაგეგმვის მექანიზმი განაწილებულია ოპერაციულ სისტემასა და გამოყენებით პროგრამებს შორის. ოპერაციული სისტემიდან მართვის უფლების მიღების შემდეგ გამოყენებითი პროგრამა თვითონ განსაზღვრავს საკუთარი მორიგი იტერაციის დასრულების მომენტს და გარკვეული სისტემური გამოძახების დახმარებით მა-

რთვას უბრუნებს ოპერაციულ სისტემას; ეს უკანასკნელი წარმოქნის ამოცანების რიგს და პრიორიტეტის შესაბამისად რიგიდან შესარულებლად ირჩევს მორიგ პროგრამას.

■ **არაგამძევებადი მრავალამოცანურობის დროს** შეიძლება წარმოიშვას სიტუაცია, როდესაც აქტიური პროცესი დიდ დროს ანდომებს რაიმე სამუშაოს შესრულებას, მაგალითად, დისკის დაფორმატებას. მომხმარებელს არ შეუძლია ეს პროცესი გადაიყვანოს ფონურ რეჟიმში და სისტემა გადართოს სხვა პროცესზე, მაგალითად, ტექსტურ რედაქტირებაზე. ეს ამცირებს სისტემის გამტარობის უნარს. მის ასამაღლებლად არაგამძევებადი ოპერაციული სისტემის დროს დამპროგრამებელმა უნდა„შეითავსოს“ ცენტრალური დამგეგმავის ფუნქციებიც და შეადგინოს პროცესის მცირე „პორციებად“ შემსრულებელი პროგრამა. ზემოთ განხილულ შემთხვევაში დისკის მთლიანად დაფორმატების პროგრამას უნდა ჰქონდეს ცალკეული ბილიკების დამფორმატებელი პროგრამების ერთობლიობის სახე. თითოეული ბილიკის დაფორმატების შემდეგ პროგრამამ მართვა უნდა გადასცეს სხვა პროგრამას და მხოლოდ მისი შესრულების შემდეგ გადავიდეს მომდევნო ბილიკის დაფორმატებაზე.

■ დამპროგრამებელზე ცენტრალური დამგეგმავის ფუნქციის შესრულების დაკისრება არაგამძევებადი ***მრავალამოცანურობის***ს ნაკლიცაა და ღირსებაც. ***ნაკლია*** იმიტომ, რომ იგი მნიშვნელოვანწილად ართულებს დამპროგრამებლის სამუშაოს და ამაღლებს დამპროგრამებლის კვალიფიკაციისადმი წაყენებულ მოთხოვნებს: დაბალი და საშუალო დონის დამპროგრამებელი ვერ შეასრულებს აღნიშნულ სამუშაოს, ამისათვის მხოლოდ მაღალკვალიფიციური დამპროგრამებელია საჭირო; ***ღირსებაა*** იმიტომ, რომ დამპროგრამებელს ამოცანათა კონკრეტული ნაკრებისათვის დაგეგმვის ყველაზე „ხელსაყრელი“ ალგორითმის დამუშავების შესაძლებლობა ეძლევა, ვინაიდან იგი პროგრამაში თვითონ განსაზღვრავს პროცესიდან პროცესზე მართვის გადაცემის მომენტს, რის მეოხებითაც შეუძლია გამორიცხოს „მოუხერხებელ“ მომენტებში პროცესების შეწყვეტის შემთხვევები. არაგამძევებადი მრავალამოცანურობის ***მეორე ღირსებაა*** ის, რომ მასში უმტკივნეულადაა გადაწყვეტილი სისტემების მიერ მონაცემების ერთობლივად გამოყენების პრობლემა: პროცესი თითოეული იტერა-

ციის განმავლობაში მონაცემებს მონოპოლურად იყენებს და ამ პერიოდის განმავლობაში მას ამაში ხელს ვერავინ უშლის. ***არაგამძვებადი მრავალამოცანურობის ეფექტურად*** გამოყენების მაგალითია ***NetWare***.

გამძვებად მრავალამოცანიან სისტემებში ზემოთ აღწერილის მსგავსი სიტუაციები, როგორც წესი, არ წარმოიშობა, რადგან ცენტრალური დამპროგრამებელი „დაკიდებული“(ხანგრძლივად მომუშავე) ამოცანის შესრულებას წყვეტს. გარდა ამისა, ***გამძვებადი სისტემების*** მნიშვნელოვანეს უპირატესობას წარმოადგენს ერთი ამოცანიდან მეორეზე გადართვის უფრო მაღალი სიჩქარე. ყველა თანამედროვე ოპერაციული სისტემა (***UNIX, Windows NT, OS/2, VAX/ /VMS***) იყენებს ***გამძვებად მრავალამოცანურობას***. შეიძლება ამის გამო ხშირად ასეთ მრავალამოცანურობას ***ჭეშმარიტ მრავალამოცანურობასაც უწოდებენ.***

1.4. პროცესთა „თანაცხოვრების“ პრობლემის გადაწყვეტის გზები

კომპიუტერულ სისტემაში ერთდროულად საკმაოდ დიდი რაოდენობის პროცესები შეიძლება მიმდინარეობდეს, ე. ი. მასში არსებობს მრავალსუბიექტიანი გარემო. ასეთ გარემოში სუბიექტების გზები აუცილებლად გადაიკვეთება და მოხდება მათ შორის გარკვეული სახის ურთიერთზემოქმედება. ამ ზემოქმედებებმა არ უნდა დაარღვიოს სუბიექტების (ე. ი. პროცესების) ნორმალური „ცხოვრება“და მათ ხელი არ უნდა შეუშალოს საკუთარი მისიის წარმატებით შესრულებაში. სხვა სიტყვებით რომ ვთქვათ, უზრუნველყოფილი უნდა იყოს პროცესების ნორმალური „თანაცხოვრება“, რაც სისტემების შემქმნელებისაგან გარკვეულ ძალისხმევას მოითხოვს. განვიხილოთ ამ ძალისხმევის შედეგად დამუშავებულ თანაცხოვრების რამდენიმე წესი, რომელთა გაცნობა გააფართოებს ჩვენს თვალსაწიერს და დაგვეხმარება კომპიუტერულ სისტემაში არსებული სიტუაციის ნათლად გააზრებაში.

მონაცემების გასაცვლელად, ერთსა და იმავე ფაილიდან მონაცემების ასაღებად და სხვა მსგავსი ფუნქციების შესასრულებლად

მულტიდაპროგრამების რეჟიმში მიმდინარე პროცესები ერთმანეთთან გარკვეული სახის ურთიერთობებს ამყარებს. ამ დროს აუცილებელია ისინი მოქმედებდეს ურთიერთშეთანხმებულად (სინქრონულად). შეუთანხმებელმა (ასინქრონულმა) ქმედებებმა შეიძლება მათი არასწორი მუშაობა და სისტემის კრახის გამოიწვიოს.

ზემოთ აღნიშნულის ილუსტრირებისათვის განვიხილოთ ფაილების ბეჭდვის (***პრინტსერვერის***) პროგრამის მაგალითი. ეს პროგრამა რიგრიგობით ბეჭდავს ყველა ფაილს, რომელთა სახელებს შემოსვლის კვალობაზე საყოველთაოდ ხელმისაწვდომ ფაილ ***„შეკვეთებში“*** ჩაწერს სხვა პროგრამები. ამ ფაილის პირველი თავისუფალი პოზიციის ნომერს, რომელშიც უნდა ჩაიწეროს დასაბეჭდი ფაილის სახელი, შეიცავს სპეციალური ცვლადი ***NEXT***, რომელიც ასევე ხელმისაწვდომია ყველა კლიენტი პროცესისათვის. შემოსულმა კლიენტმა პროცესმა უნდა წაიკითხოს ***NEXT*** ცვლადის მნიშვნელობა, ფაილ „შეკვეთების“ შესაბამის პოზიციაზე ჩაწეროს დასაბეჭდი ფაილის სახელი და ***1*** ერთეულით გაზარდოს ***NEXT***–ის მნიშვნელობა.

*დავუშვათ, რომ რომელიღაც **L** პროცესმა წაიკითხა, რომ **NEXT**–ის მნიშვნელობაა **7**, დაიმახსოვრა იგი და გარკვეული მიზეზის (მაგალითად, გამოყოფილი კვანტის) ამოწურვის გამო ვერ მოასწრო ფაილ «შეკვეთების» მე-**7** პოზიციაზე ჩაეწერა დასაბეჭდი ფაილის სახელი. ამის გამო **NEXT**–ის მნიშვნელობა ვერ გაიზარდა და იგი კვლავ **7**-ის ტოლი დარჩა. მომდევნო **Q** პროცესმა ეს ნომერი წაიკითხა, ფაილ „შეკვეთების“ მე-**7** პოზიციაზე ჩაწერა დასაბეჭდი ფაილის სახელი და **NEXT**–ის მნიშვნელობა **8**-ის ტოლი გახადა.*

*შემდგომში, როდესაცმართვა ზემოთ აღნიშნულ **L** პროცესს გადაეცემა, იგი არ წაიკითხავს **NEXT**–ის მნიშვნელობას და საკუთარი დასაბეჭდი ფაილის სახელს ჩაწერს მის მის მიერ დამახსოვრებულ მე-**7** პოზიციაზე. ამ უკანასკნელ პოზიციაზე **Q** პროცესის მიერ ჩაწერილი ფაილის სახელი წაიშლება და ეს ფაილი ვერ ელირსება დაბეჭდვას.*

ზემოთ მოყვანილ მაგალითში ***L***, ***Q*** პროცესები თითქოსდა ერთმანეთს „შეეჯიბრა“და მეორემ „გადაასწრო“პირველს; აღნიშნულ-ის გამო ამ მოვლენას ***„შეჯიბრი“*** ეწოდა. მოცემულ შემთხვევაში ***L*** და ***Q*** პროცესებს შორის შეჯიბრის მოვლენა იმიტომ წარმოიშვა, რომ ორივე მათგანი იყენებდა ერთსა და იმავე ***NEXT*** ცვლადს. რამდენიმე პროცესის მიერ გამოყენებად ცვლადებს, მონაცემებსა თუ რესურსებს შესაბამისად ამ პროცესებს შორის ***განაწილებადი ცვლადები, მონაცემები და რესურსები***, ხოლო მათი დასაკუთრების მსურველ პროცესებს – ***კონკურენტი პროცესები*** ვუწოდოთ.

შეჯიბრის დროს კონკურენტი პროცესებს შორის ირღვევა ურთიერთშეთანხმებული (სინქრონული) მუშაობა, თავად შეჯიბრს კი წარმოშობილი სიტუაციების ***არარეგულარულობა*** იწვევს. რეგულარულობის აღდგენით პროცესების ფუნქციონირებაში გამოირიცხება შეჯიბრის წარმოშობის შესაძლებლობა და კონკურენტი პროცესები ***ურთიერთშეთანხმებულ პროცესებად*** გადიქცევა.

პროცესების სინქრონიზებისათვის უმნიშვნელოვანესია ე. წ. „კრიტიკული სექციის“ ცნება.

კრიტიკული სექცია ეწოდება პროგრამის იმ ნაწილს, რომლიდანაც უშუალოდ ხდება გასანაწილებელ ერთეულებზე (ცვლადებზე, მონაცემებზე, რესურსებზე) გადასვლა. აღნიშნულიდან გამომდინარე, კრიტიკული სექცია უშუალოდ ემიჯნება გასანაწილებელ ერთეულს (ცვლადს, მონაცემს, რესურსს).

შეჯიბრის წარმოშობის შესაძლებლობა გამოირიცხება, თუ დროის ნებისმიერ მომენტში გასანაწილებელი ერთეულის მეზობელ კრიტიკულ სექციაში ერთ პროცესზე მეტი არ იქნება. ***კონკურენტი პროცესების ურთიერთგამორიცხვა*** ნიშნავს კრიტიკულ სექციაში მათი ერთდროულად შესვლის გამორიცხვას (ამ სექციაში მათი რიგრიგობით დაშვების უზრუნველყოფას). არსებობს კონკურენტი პროცესების გამორიცხვის შემდეგი ორი ხერხი.

ურთიერთგამორიცხვის უმარტივესი ხერხიაკრიტიკულ სექციაში არსებული პროცესისათვის ყველა შეწყვეტის აკრძალვის უფლების მიცემა. ამ ხერხის გამოყენება არ ვარგა, რადგან სახიფათოა პროცესს ვანდოთ სისტემის მართვა. მან, ***ჯერ ერთი,*** შეიძლება დიდი ხნით დაიკავოს (მოაცდინოს) პროცესორი და, ***მეორე მხრივ,*** კრიტიკულ სექციაში არსებული პროცესის კრახი მთელი სისტემის კრახს გამოიწვევს, რადგან შეწყვეტები არასდროს იქნება ნებადართული.

უმჯობესია გამოვიყენოთ მაბლოკირებელი ცვლადები. ამისათვის გასანაწილებელ რესურსს უნდა შევუთანადოთ ორობითი ცვლადი, რომელიც ***1***-ის ტოლ მნიშვნელობას მიიღებს რესურსის თავისუფლების დროს (კრიტიკულ სექციაში არც ერთი პროცესის არ არსებობისას) და ***0***-ის ტოლ მნიშვნელობას – საწინააღმდეგო შემთხვევაში.

ყველა პროცესის ზემოთ აღწერილი შეთანხმების გამოყენებით ორგანიზებისას ურთიერთგამორიცხვა გარანტირებული იქნება. უნდა შევნიშნოთ, რომ ***ერთდროულად უნდა შესრულდეს მაბლოკირებელი***

ცვლადის შემოწმებისა და მისი დაყენების ოპერაციები. დავუშვათ, რომ ცვლადის შემოწმებით პროცესმა დაადგინა, რომ რესურსი თავისუფალია, მაგრამ იმავდროულად ცვლადს ვერ მისცა ***0***-ის ტოლი მნიშვნელობა და იგი შეჩერდა კვანტის ამოწურვის გამო. მისი შეჩერების განმავლობაში დავუშვათ, რომ მეორე პროცესმა დაიკავა რესურსი, შევიდა თავის კრიტიკულ სექციაში და ისე შეჩერდა, რომ მიღებულ გასანაწილებელ რესურსზე მუშაობა ვერ დაამთავრა. მართვა დაუბრუნდება პირველ პროცესს; იგი შეჩერებამდე ჩატარებული შემოწმებით დარწმუნებულია, რომ რესურსი თავისუფალია და ამიტომ დააყენებს დაკავებულობის ნიშანს და შევა კრიტიკული სექციაში, რომელშიც უკვე მეორე პროცესია შესული. ამგვარად დაირღვევა ურთიერთგამორიცხვის პრინციპი, რამაც შეიძლება არასასურველი შედეგი გამოიწვიოს. სწორედ ასეთი სიტუაციების გამოსარიცხავად უნდა შესრულდეს ერთდროულად შემოწმებისა და დაყენების ოპერაციები. ამისათვის მანქანის ბრძანებათა სისტემაში უნდა არსებობდეს ***„შემოწმება-დაყენების“*** ერთიანი ბრძანება, ან სისტემური საშუალებით უნდა მოხდეს სათანადო ***პროგრამული პრიმიტივების*** რეალიზება, რომლებიც მაბლოკირებელი ორობითი ცვლადის შემოწმება-დაყენების ოპერაციების დამთავრებამდე აკრძალავს კრიტიკულ სექციებში შესვლას.

მაბლოკირებელი ხერხის გამოყენებით კრიტიკული სექციების რეალიზებას აქვს არსებითი ნაკლი: კრიტიკულ სექციაში რომელიმე პროცესის არსებობისას სხვა პროცესი, რომელსაც სჭირდება იგივე რესურსი, მაბლოკირებელი ცვლადის გამოკითხვის რუტინულ ოპერაციის შესრულებაზე ***უსარგებლოდ ფლანგავს პროცესორულ დროს.*** ასეთი სიტუაციების გამოსარიცხავად შეიძლება გამოვიყენოთ ე. წ. ***ხდომილობათა აპარატი.*** სხვადასხვა ოპერაციულ სისტემაში ხდომილობათა აპარატი სხვადასხვანაირად რეალიზდება, მაგრამ ნებისმიერ შემთხვევაში გამოიყენება ანალოგიური დანიშნულების სისტემური ფუნქციები. ისინი პირობითად აღინიშნება ტერმინებით ***WAIT(x)*** [ანუ **ლოდინი(*x*)**] და ***POST(x)*** [ანუ **მიწოდება(*x*)**], სადაც ***x*** არის გარკვეული ხდომილობის იდენტიფიკატორი.

რესურსი თუ დაკავებულია პროცესი ციკლურ გამოკითხვის ნაცვლად გამოიძახებს სისტემურ ***WAIT(D)*** ფუნქციას, სადაც ***D*** აღნიშნავს ***D*** რესურსის გათავისუფლების ხდომილობას. ***WAIT(D)***

ფუნქციას აქტიური პროცესი გადაჰყავს რესურსის გათავისუფლების ლოდინის მდგომარეობაში და დესკრიპტორში აკეთებს პროცესის მიერ ***D*** რესურსის ლოდინის მაფიქსირებელ ნიშნულს. ლოდინის პერიოდში ***D*** რესურსის გამომყენებელი პროცესი კრიტიკული სექციდან გამოსვლის შემდეგ შეასრულებს ***POST(D)*** ფუნქციას, კერძოდ, ოპერაციული სისტემა გააქტიურებს მზა პროცესების რიგში მდგარ იმ პროცესს, რომელიც ***D*** ხდომილობას (ჩვენ შემთხვევაში - ***D*** რესურსის გათავისუფლებას) ელოდებოდა.

ნიდერლანდენმა მეცნიერმა, სტრუქტურული დაპროგრამების ერთერთმა შემქმნელმა, ინფორმატიკაში ყველაზე პრესტიჟული ტიურინგისეული პრემიის ლაურეატმა ***ე. დეიკსტრამ*** (***1930-2002***) პროცესების სინქრონიზაციის განსაზოგადებლად შემოიტანა პროგრამული ***P*** და ***V*** პრიმიტივები, რომლებიც ოპერაციებს ასრულებს ***სემაფორებად*** წოდებულ მთელ არაუარყოფით ცვლადებზე. იგი აღვნიშნოთ ***S*** სიმბოლოთი (***S****emaphore*). მასზე ჩასატარებელი ოპერაციები ასე განისაზღვრება:

- ***V(S)*** ოპერაცია ერთიანი მოქმედებით ახდენს ***S*** ცვლადის ინკრემენტირებას; ამორჩევის, ინკრემენტისა და დამახსოვრების პროცესები მიმდევრობით შეწყვეტის გარეშე უნდა შესრულდეს;
- ***P(S)*** ოპერაცია შესაძლებლობის შემთხვევაში ახდენს ***S*** ცვლადის დეკრემინტირებას. ***S=0***-ის დროს ***S***-ის დეკრემიენტირება ვერ შესრულდება, რადგან იგი არ გამოდის მთელი არაუარყოფითი მნიშვნელობათა არედან. ამ შემთხვევაში ***P***-ოპერაციის გამომწვევი პროცესი ელოდება იმ მომენტს, როდესაც დეკრემენტირება შესაძლებელი გახდება. წარმატებული შემოწმებისა და შემცირებისოპერაციები ერთი განუყოფადი ოპერაციაა.

კერძო შემთხვევაში, როდესაც ***S*** სემაფორი მხოლოდ ***0***-ის ან ***1***-ის ტოლ მნიშვნელობებს იღებს, იგი მაბლოკირებელ ცვლადად გარდაიქმნება. ***P*** ოპერაციას შეუძლია ლოდინის რეჟიმში გადაიყვანოს პროცესი, ხოლო ***V*** ოპერაციას - გარკვეულ პირობების არსებობის შემთხვევაში გააქტიუროს ***P*** ოპერაციის მიერ შეჩერებული სხვა პროცესი (ეს ოპერაციები ჩამოჰგავს ზემოთ განხილულ ***WAIT*** და ***POST*** ოპერაციებს).

სემაფორების ნაკლოვანებებია: ***1)*** ისინი იმდენად ელემენტარულებია, რომ რთულ პარალელურ გამოთვლებს მარტივად ვერ აღწერს;

2) მათი გამოყენებისას რთულდება კორექტურობის დამტკიცებას; ***3)*** მათმა არასწორად გამოყენებამ შეიძლება დაარღვიოს როგორც პროგრამის, ისე მთელი სისტემის მუშაობის უნარი.

შევეხოთ პროცესების სინქრონიზაციის დროს წარმოშობილ ერთ სპეციფიკურ სიტუაციას, რომელსაც ***კლინჩი*** ეწოდება. ეს ტერმინი შემოვიდა კრივიდან, სადაც ***კლინჩი*** (ინგ. *clinch*) ეწოდება სპორტსმენის დაცვით მოქმედებას, როდესაც იგი შემტევ მოწინააღმდეგეს ხელების „შებოჭვის“გზით უზღუდავს მოძრაობის უნარს ინფორმატიკაში ***კლინჩი ეწოდება*** ჩიხურ სიტუაციას, რომლის დროსაც პროცესები ერთმანეთის დასრულების ლოდინის რეჟიმშია.

განვიხილოთ კლინჩის, ანუ ჩიხური სიტუაციის მაგალითი. დავუშვათ, რომ მულტიდაპროგრამების რეჟიმში მიმდინარე ***A*** და ***B*** პროცესებს მუშაობისათვის სჭირდება ორი რესურსი: პრინტერი და დისკი. დავუშვათ, რომ ***A*** პროცესმა დაიკავა პრინტერი (დააყენა მაბლოკირებელი ცვლადი) და შეწყდა; მართვას აიღებს ***B*** პროცესი და დავუშვათ, პირველად დაიკავა დისკი; ამის შემდეგ იგი ვერ გაგრძელდება, რადგან მისთვის სასურველი მეორე რესურსი დაკავებული აქვს ***A*** პროცესს; ამიტომ იგი გაჩერდება და მართვა დაუბრუნდება ***B*** პროცესს. აქ წარმოიშვება ***კლინჩური სიტუაცია***: ***A*** პროცესს დაკავებული აქვს ***B*** პროცესისათვის საჭირო რესურსი (პრინტერი), ხოლო ***B*** პროცესს - ***A*** პროცესისათვის საჭირო რესურსი (დისკი). ასეთ მდგომარეობაში ***A*** და ***B*** პროცესები შეიძლება დიდი ხნის განმავლობაში დარჩეს.

საკუთარ სიჩქარეებს შორის არსებულ თანაფარდობებზე დამოკიდებულებით პროცესებმა შეიძლება ერთმანეთისაგან დამოუკიდებლად დაიკავოს საჭირო რესურსები, ან წარმოქმნას რიგი ამ რესურსების დასაკავებლად, ან მოახდინოს ერთმანეთის ბლოკირება. ***ჩიხური სიტუაციები*** უნდა განვასხვაოთ ***მარტივი რიგებისაგან***, თუმცა ორივე წარმოიშობა პროცესების მიერ რესურსების ერთობლივად გამოყენების შემთხვევაში და გარეგნულად ერთმანეთს ჰგავს: პროცესი ჩერდება და ელოდება რესურსის განთავისუფლებას. ოღონდ რიგი ნორმალური მდგომარეობაა, რომელიც შეკვეთების შემთხვევით შემოსვლის დროს რესურსების გამოყენების მაღალი კოეფიციენტის განუყრელი ნიშანია. იგი მაშინ წარმოიშობა, როდესაც რესურსი მოცემულ მომენტშია მიულწეველი, თორემ გარკვეული დროის შემდეგ იგი გათავი-

სუფლდება და პროცესის შესრულება გაგრძელდება. სახელწობიდან გამომდინარე, ***ჩიხი*** გადაუწყვეტი სიტუაციაა.

განხილულ შემთხვევაში ჩიხი ორმა პროცესმა წარმოქმნა, თუმცა ერთმანეთი შეიძლება რამდენიმე პროცესმაც დაბლოკოს.

ჩიხების პრობლემა წარმოშობს ჩიხების თავიდან აცილების, ჩიხების ამოცნობისა და ჩიხების წარმოქმნის შემდეგ სისტემის აღდგენის ამოცანებს.

ჩიხები შეიძლება თავიდან ავიცილოთ პროგრამების დაწერის, სტადიაზე, ე. ი. პროგრამები ისე უნდა დავწეროთ, რომ პროცესების სიჩქარეთა ნებისმიერი თანაფარდობის დროს ჩიხები ვერ წარმოიქმნას. მაგალითად, ზემოთ განხილულ მაგალითში ***A*** და ***B*** პროცესებს რესურსები ერთნაირი თანამიმდევრობით რომ მოეთხოვნათ, ჩიხი პრინციპულად არ წარმოიქმნებოდა. ჩიხების თავიდან აცილების მეორე მიდგომას ***დინამიკური მიდგომა*** ეწოდება; მის შემთხვევაში გამოიყენება პროცესებისათვის რესურსების გამოყოფის გარკვეული წესები; მაგალითად, რესურსები შეიძლება გამოიყოს გარკვეული მიმდევრობის დაცვით, რომელიც საერთო იქნება ყველა პროცესისათვის.

ზოგიერთ შემთხვევაში, როდესაც ჩიხური სიტუაცია ბევრი რესურსის გამომყენებელი მრავალი პროცესითაა წარმოქმნილი, ჩიხის ამოცნობის ამოცანა არატრივიალურია. არსებობს ჩიხების ამოცნობის პროგრამულად რეალიზებული ფორმალური მეთოდები, რომლებიც დაფუძნებულია რესურსების განაწილების ცხრილებისა და დაკავებულ რესურსების მოთხოვნათა ცხრილების გამოყენებაზე. ამ ცხრილების ანალიზი ურთიერთბლოკირების აღმოჩენის საშუალებას გვაძლევს.

ჩიხური სიტუაციის წარმოშობისას არაა აუცილებელი შესრულებიდან მოვხსნათ ყველა დაბლოკილი პროცესი. შეიძლება: ***1)***მოვხსნათ მათი მხოლოდ ნაწილი, რის შედეგაც დანარჩენ პროცესებს გაუთავისუფლდებათ რესურსები; ***2)*** ზოგიერთი პროცესი დავაბრუნოთ სვოპინგის (იხ.გვ. 50) არეში; ***3)*** ზოგიერთი პროცესი „გადავაგოროთ“ ე.წ საკონტროლო წერტილამდე, რომელშიც ხდება მოცემული წერტილიდან პროცესის გასაგრძელებლად საჭირო ინფორმაციის დამახსოვრება; ***საკონტროლო წერტილები*** პროგრამაში იმ წერტილებზე განთავსდება, რომლებიდანაც შეიძლება წარმოიშვას ჩიხები.

სემაფორები ძალიან ფრთხილად უნდა გამოვიყენოთ, რადგან ერთმა უმნიშვნელი შეცდომამ შეიძლება გააჩეროს სისტემა. კორექტული პროგრამების დაწერის გაიოლებისათვის შემოთავაზებული იქნა მონიტორად წოდებული სინქრონიზების მაღალი დონის საშუალება.

მონიტორი პარალელიზმის ორგანიზების მექანიზმია, რომელიც შეიცავს კონკრეტული საერთო რესურსის დინამიკურად განაწილებისათვის აუცილებელ როგორც მონაცემებს, ისე პროცესებს. იგი პროცესების, ცვლადებისა და მონაცემთა სტრუქტურების ნაკრებია. პროცესებს შეუძლია გამოიწვიოს მონიტორის პროცედურები, მაგრამ მონიტორის შინაგან მონაცემებთან შეღწევა არ შეუძლია. მონიტორებს აქვს მნიშვნელოვანი თვისება, რომლის ძალითაც ისინი სასარგებლო ხდება ურთიერთგამორიცხვის მისაღწევად: ***მონიტორის მიმართ აქტიური შეიძლება იყოს მხოლოდ ერთი პროცესი.*** მონიტორის პროცედურათა გამოძახებებს კომპილატორი სპეციალური სახით ამუშავებს. ჩვეულებრივ, როდესაც პროცესი გამოიძახებს მონიტორის პროცედურას, ამ პროცედურის პირველი რამდენიმე ბრძანება ამოწმებს მოცემული მონიტორის მიმართ აქტიურია თუ არა რომელიმე სხვა პროცესი. დადებითი პასუხის შემთხვევაში გამომძახებელი პროცესი მანამ ჩერდება, სანამ აღმოჩენილი აქტიური პროცესი არ გაათავისუფლებს მონიტორს. ამგვარად, მონიტორში რამდენიმე პროცესის შესვლას გამოირცხავს არა დამპროგრამებლი, არამედ კომპილატორი, რაც ამცირებს შეცდომის წარმოშობის ალბათობას.

1.5. ოპერაციული სისტემის მიერ მეხსიერების მართვის ძირითადი საკითხები

მეხსიერება უმნიშვნელოვანესი რესურსია, რომელიც საგულდაგულოდ უნდა მართოს მულტიპროგრამულმა ოპერაციულმა სისტემამ. გასანაწილებელია ოპერაციული სისტემის მიერ დაუკავებელი მთელი ოპერატიული მეხსიერება. ოპერაციული სისტემა ჩვეულებრივ თავსდება ოპერატიული მეხსიერების უმცროსი მისამართებით წარმოშობილ სივრცეზე, თუმცა შესაძლებელია იგი უდიდესი მისამართებით წარმოშობილ სივრცეშიც განთავსდეს. მეხსიერების მართვისათვის ოპერაციულმა სისტემამ უნდა შეასრულოს შემდეგი ფუნქციები: თვალყური ადევნოს თავისუფალ და დაკავებულ ოპერატიულ მეხსი-

ერებას; პროცესებს გამოუყოს ოპერატიული სივრცის უბნები და ისინი გაათავისუფლოს პროცესის დამთავრების შემდეგ; პროცესები თუ ვერ ეტევა ოპერატიულ მეხსიერებაში, მაშინ ზოგიერთი მათგანი გაიტანოს დისკზე და ისინი ოპერატიულ მეხსიერებაში თანდათანობით გადაიტანოს აღნიშნული მეხსიერების გათავისუფლების კვალობაზე; პროგრამის მისამართები უნდა შეუთანადოს ფიზიკური მეხსიერების კონკრეტულ უბანს.

ცვლადებისა და ბრძანებების იდენტიფიცირენისათვის გამოიყენება სიმბოლური სახელები (ნიშნები), ვირტუალური მისამართები და ფიზიკური მისამართები.

სიმბოლურ სახელებს მომხმარებელი ცვლადებსა და ბრძანებებს ანიჭებს ალგორითმულ ენაზე ან ასემბლერზე პროგრამის დაწერის დროს.

ვირტუალურ მისამართებს გამოიმუშავებს სამანქანო ენაზე პროგრამის მთარგმნელი ტრანსლატორი. ვინაიდან ტრანსლაციის დროს ზოგადად უცნობია პროგრამა ოპერატიული მეხსიერების თუ რომელ ადგილზე იქნება ჩატვირთული, ამიტომ ტრანსლატორი ცვლადებსა და ბრძანებებს ანიჭებს ***ვირტუალურ (წარმოსახვით) მისამართებს;*** ამასთანავე პროგრამის განთავსება უსიტყვოდ იწყება უჯრედიდან, რომლის მისამართია ***0***. პროცესის ვირტუალური მისამართების ერთობლიობას ეწოდება ***ვირტუალური სამისამართო სივრცე.*** თითოეულ სივრცეს საკუთარი ვირტუალური სივრცე აქვს. ვირტუალუ-რი სამისამართო სივრცის ზომა იზღუდება მისამართის თანრიგების რაოდენობით, რომელიც შეიძლება ჰქონდეს მოცემული არქიტექტურის კომპიუტერს. იგი, როგორც წესი, არ ემთხვევა კომპიუტერში არსებული მფზიკური მეხსიერების მოცულობას.

ფიზიკურ მისამართებს წარმოქმნის ოპერატიული მეხსიერების იმ უჯრდთა ნომრები, რომლებშიც ცვლადებისა და ბრძანებების განთავსებაა შესაძლებელი. ვირტუალური მისამართებიდან ფიზიკურ მისამართებზე გადასვლა შეიძლევა ორი ხერხით მოხდეს. ***პირველი ხერხის*** დროს ფიზიკურ მისამართებად ვირტუალურ მისამართებს ცვლის სპეციალური სისტემური პროგრამა – ***გადამწაცვლებელი ჩამტვირთავი.*** ამ უკანასკნელს აქვს საწყისი მონაცემები იმ ფიზიკური მეხსიერების საწყისი მისამართის შესახებ, რომელშიც უნდა ჩაიტვირთოს პროგრამა; ტრანსლატორი ***გადამწაცვლებელ ჩამტვირ-***

თავს აწვდის ინფორმაციას პროგრამის დასამისამართებადი კონსტანტების შესახებ. საკუთარი საწყისი მონაცემებისა და ტრანსლატორიდან მიღებული ინფორმაციის საფუძველზე აღნიშნული ჩამტვირთავი ოპერატიულ მეხსიერების ცარიელ ადგილზე ჩატვირთავს პროგრამას, რომლის დროსაც ვირტუალურს ფიზიკური მისამართებით ცვლის.

მეორე ხერხის დროს პროგრამის კომპონენტები ოპერატიული მეხსიერების უჯრედებში მათ ვირტუალურ მისამართებთან ერთად ჩაიტვირთება, ხოლო ოპერაციული სისტემა აფიქსირებს პროგრამის ვირტუალურ და რეალურ სამისამართო სივრცეებს შორის არსებულ ძვრას. პროგრამის შესრულებისას ოპერატიულ მეხსიერებასთან ყოველი მიმართვის დროს ვირტუალური მისამართი გარდაიქმნება ფიზიკურ მისამართად.

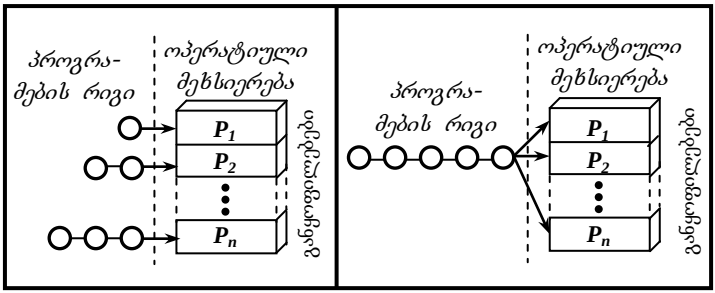
პირველი ხერხის ნაკლია მისი სიხისტე: გადამნაცვლებელი ჩამტვირთავი პროგრამას ხისტად მიაჯაჭვავს მისი განთავსების ადგილთან, რის გამოც შეუძლებელი ხდება პროგრამას შეეცვალოს ადგილმდებარეობა მისი რეალიზების დროს. ***ღირსებაა*** მაღალი მწარმოებლურობა: ნებისმიერი ვირტუალური მისამართი მხოლოდ ერთხელ - ოპერატიულ მეხსიერებაში პროგრამის ჩატვირთვისას გარდაიქმნება. ***მეორე ხერხისღირსებაა*** მოქნილობა: პროგრამას შესრულების პერიოდშიც შეიძლება შეეცვალოს ადგილმდებარეობა, ***ნაკლია*** შედარებით ნაკლები მწარმოებლურობა, რასაც იწვევს მისართთან ყოველი მიმართვის დროს ფიზიკურ მისამართად ვირტუალური მისამართის გარდაქმნის გამო წარმოშობილი ზედნადები ხარჯები.

ზოგიერთ შემთხვევებში (ჩვეულებრივ სპეციალიზებულ სისტემებში), როდესაც წინასწარ ზუსტადაა ცნობილი ოპერატიული მეხსიერების თუ რომელ ნაწილშია გათვალისწინებული პროგრამის შესრულება, ტრანსლატორი ერთბაშად გასცემს ფიზიკური მისამართებით დამისამართებულ კოდს.

მეხსიერების მართვის მეთოდები ორ კლასად შეიძლება დავყოთ: მეთოდებად, რომლებიც პროცესებს გადაანაცვლებს ოპერატიულ მეხსიერებასა და დისკს შორის და მეთოდებად, რომლებიც ამას არ აკეთებს. მარტივია უკანასკნელი მეთოდები და პირველად ისინი განვიხილოთ.

I.1. მეხსიერების განაწილება ფიქსირებული სიდიდის განყოფილებებად

ოპერატიული მეხსიერების მართვის უმარტივესი ხერხია მისი დაყოფა ფიქსირებული სიდიდის რამდენიმე განყოფილებად. იგი შეიძლება სისტემის სტარტიის ან მისი გენერირების დროს ხელით მოახდინოს ოპერატორმა. შესასრულებლად მოსულ მორიგ ოპერაცია ადგილს იკავებს რომელიმე განყოფილებაში შემსვლელი ოპერაციების რიგში ან საერთო რიგში (***ნახ.1.2***)



ნახ.1.2. *მეხსიერების განაწილების ხერხები*

მეხსიერების მართვის ქვესისტემაამ შემთხვევაში: შესასრულებლად მოსული პროგრამის ზომას უდარებს თავისუფალ განყოფილებათა ზომებს; შეირჩევს შესაფერისი ზომის განყოფილებას; მასში ჩატვირთავს პროგრამას და ააწყობს მის მისამართებს.

მოცემული მეთოდის ღირსებაა – ***რეალიზების სიმარტივე***, ნაკლი კი – ***სიხისტე***. თითოეულ განყოფილებაში რადგან მხოლოდ ერთი პროგრამა შეიძლება შესრულდეს, ამიტომ ***მულტიდაპროგრამების დონე*** პროგრამის ზომისაგან დამოუკიდებლად წინასწარ იზღუდება განყოფილობათა რაოდენობით. რაგინდ მცირე არ უნდა იყოს პროგრამის მოცულობა, იგი მაინც ერთ განყოფილებას იკავებს, რაც იწვევს მეხსიერების არაეფექტურ გამოყენებას: ოპერატიული მეხსიერების თავისუფალი ადგილების საერთო მოცულობა საკმარისიც რომ იყოს პროგრამის შესასრულებლად, ამას შეუძლებელს ხდის მეხსიერების განყოფილებებად დაყოფა.

I.2. მეხსიერების განაწილება ცვლადი სიდიდის განყოფილებებად

ამ შემთხვევაში მესიერება წინასწარ არ იყოფა განყოფილებებად. თავდაპირველად თავისუფალია მთელი მეხსიერება. თითოეულ მოსულ ამოცანას მეხსიერებაში გამოეყოფა მისთვის აუცილებელი ზომის ადგილი. მეხსიერების თავისუფალი ადგილის ზომა საკმარისი თუ არ აღმოჩნდება ამოცანისათვის, მაშინ ეს უკანასკნელი შესასრულებლად არ მიიღება და რიგში ჩადგება. მეხსიერებაში არსებული ამოცანის შესრულების შემდეგ გაიზრდება მისი თავისუფალი სივრცე და მასში ჩაიტვირთება რიგში მდგარი ამოცანა. ამგვარად, დროის ნებისმიერ მომენტში ოპერატიული მეხსიერება შედგება წინასწარ განუსაზღვრელი ზომის დაკავებული და თავისუფალი უბნების შემთხვევითი მიმდვრობისაგან. მეხსიერების მართვის მოცემული მეთოდის დროს ***ოპერაციული სისტემის ამოცანებია***:

- ოპერატიული მეხსიერების თავისუფალი და დაკავებული უბნების ცხრილების შედგენა, რომლებშიც მიეთითება უბნების საწყისი მისამართები და მოცულობები;
- ახალი ამოცანის მოსვლისას – მოთხოვნის გაანალიზება, თავისუფალ უბანთა ცხრილის დათვალიერება და ისეთი განყოფილების ამორჩევა, რომელშიც მოთავსდება შემოსული ამოცანა.
- ამორჩეულ განყოფილებაში ამოცანის ჩატვირთვა; მეხსიერების თავისუფალი და დაკავებული სფეროთა ცხრილის კორექტირება;
- ამოცანის დასრულების შემდეგ მეხსიერების თავისუფალი და დაკავებული სფეროთა ცხრილის ხელახალი კორექტირება;

შესრულების პროცესში პროგრამული კოდი არ გადანაცვლდება, ე. ი. გადანაცვლებადი ჩამტვირთველის გამოყენებით შეიძლება მისამართების ერთდროულად აწყობა.

ახლად შემოსული ამოცანისათვის განყოფილება სხვადასხვა წესით შეიძლება ამოირჩეს; მაგალითად შეიძლება ამოირჩეს მინიმალრად საკმარისი, ან მაქსიმალურად საკმარისი ზომის განყოფილება და ა. შ. ნებისმიერ ამ წესს აქვს როგორც ღირსებები, ისე ნაკლოვანებებიც.

ფიქსირებული ზომის განყოფილებებად მეხსიერების განაწილების მეთოდთან შედარებით მოცემულ მეთოდი გაცილებით მოქნილია, მაგრამ აქვს სერიოზული ნაკლიც – მეხსიერების ფრაგმენტირება. ***ფრა-***

გმენტირება ეწოდება მეხსიერებაში ძალიან მცირე ზომის თავისუფალი არამომიჯნავე უბნების (ფრაგმენტების) წარმოქმნას. ცალკე აღებული ფრაგმენტები იმდენად მცირეა, რომ მასში ვერ თავსდება ვერც ერთი მოსული პროგრამა, მაგრამ ფრაგმენტების ჯამური მოცულობა საკმარისია ნებისმიერი მოსული პროგრამისათვის.

ფრაგმენტირების წინააღმდეგ ბრძოლის ერთ-ერთი მეთოდია ერთიანი თავისუფალი არის წარმოსაქმნელად მეხსიერების ყველა დაკავებული უბნის გადანაცვლება უფროსი ან უმცროსი მისამართებისაკენ. ეს მოითხოვს ცვლადი სიდიდის განყოფილებებიანი მეხსიერების მართვისათვის ოპერაციული სისტემის მიერ შესასრულებელ ზემოთ ჩამოთვლილ ფუნქციებს დაემატოს ერთ-ერთი განყოფილების შიგთავსის მეორე განყოფილებაში დროდარო კოპირებისა და თავისუფალი და დაკავებული უბანთა ცხრილი კორექტირების პროცედურებიც. ერთიანი თავისუფალი სივრცის წარმოქმნის ზემოთ აღწერილ პროცედურას ეწოდება ***შეკუმშვა***. იგი შეიძლება განხორციელდეს ან ყოველი ამოცანის დასრულების შემდეგ, ან მაშინ, როდესაც ახლად შემოსული ამოცანისათვის ვერ მოიძებნება საჭირო ზომის თავისუფალი განყოფილება. პირველ შემთხვევაში ცხრილის კორექტირებისათვის მცირე გამოთვლითი სამუშაოს ჩატარებაა საკმარისი, ხოლო მეორე შემთხვევაში შეკუმშვის პროცედურა იშვიათადაა ჩასატარებელი. პროგრამები შესრულების პროცესში გადაადგილდება ოპერატიული მეხსიერების ფარგლებში, რის გამოც ვირტუალური მისამართები ფიზიკურ მისამართებად დინამიკურად გარდაიქმნება.

შეკუმშვა მეხსიერების უფრო უკეთესად გამოყენების საშუალებას გვაძლევს, მაგრამ მართვისათვის მოითხოვს ძალიან დიდ დროს, რაც ხშირად აუფასურებს მის ღირსებას.

II. მეხსიერების განაწილება დისკური სივრცის გამოყენებით

ოპერატიულ მეხსიერებაში ისეთი პროგრამების განსათავსებლად, რომელთა ზომა აღემატება მეხსიერებაში არსებულ ვისუფალი ადგილის ზომას, პროგრამებს ყოფენ ნაწილებად, რომლებსაც ***ოვერლეები*** (***Overlay*** - *მიერთება*) ეწოდება. ყველა ოვერლეი შეინახება დისკზე და მათ მეხსიერებასა და დისკს შორის გადაანაცვლებს ოპერატიული სისტემის საშუალებები. მეხსიერებაში შესასრულებლად პირველად გადაიგზავნება ნულოვანი ოვერლეი, რო-

მელიც დასრულებისას გამოიძახებს მომდევნო ოვერლის და ა.შ. ნაწილებად (ოვერლეებად) პროგრამა უნდა დაყოს და ოპერატიულ მეხსიერებაში მათი ჩატვირთვა უნდა დაგეგმოს დამპროგრამებელმა.

ამ მიმართულებით გამოთვლითი პროცესის ორგანიზების მეთოდების განვითარების შედეგად წარმოიქმნა ე. წ. ვირტუალური მეხსიერების მეთოდი. ***ვირტუალური ეწოდება რესურსს***, რომელიც მომხმარებელს ან სამომხმარებლო პროგრამას ისეთი თვისებების მქონე რესურსად აჩვენებს თავს, რომლებიც მას სინამდვილეში არ გააჩნია. მაგალითად, ***ვირტუალური მეხსიერება*** მომხმარებელს ისე აჩვენებს თავს, თითქოს მისი ზომა აღემატებოდეს სისტემაში არსებული რეალური ოპერატიული მეხსიერების ზომაას. მომხმარებელი ისე წერს თავის პროგრამას, თითქოს მის განკარგულებაში იყოს დიდი ზომის ოპერატიული მეხსიერება, სინამდვილეში კი პროგრამის მიერ გამოყენებული ყველა მონაცემი შენახულია ერთ ან რამდენიმე სხვადასხვა სახის დამხსომებელ მოწყობილობაში, კერძოდ, ხისტ დისკებზე, და მხოლოდ საჭიროებისამებრ ნაწილ-ნაწილ აისახება რეალურ ოპერატიულ მეხსიერებაში.

ამგვარად, ***ვირტუალური მეხსიერება*** ეწოდება პროგრამულ-აპარატურული საშუალებათა ერთობლიობას, რომელიც მომხმარებელს ოპერატიული მეხსიერების ზომაზე უფრო დიდი ზომის პროგრამების დაწერის საშუალებას აძლევს. ამისათვის ვირტუალური მეხსიერება ავტომატურად, დამპროგრამებლის მონაწილეობის გარეშე:

- მონაცემებს განათავსებს სხვადასხვა სახის დამხსომებელ მოწყობილობებში; მაგალითად, პროგრამის ერთ ნაწილს განათავსებს ოპერატიულ, ხოლო მეორე ნაწილს კი დისკურ მეხსიერებაში;
- მონაცემებს საჭიროების კვალობაზე გადაანაცვლებს სხვადასხვა ტიპის დამხსომებელ მოწყობილობებს შორის; მაგალითად, დისკიდან პროგრამის ნაწილს გადატვირთავს ოპერატიულ მეხსიერებაში;
- ვირტუალურ მისამართებს გარდაქმნის ფიზიკურ მისამართებად.

ვირტუალური მეხსიერება ყველაზე ხშირად რეალიზებულია გვერდობრივად, სეგმენტურად ან გვერდობრივ-სეგმენტურად.

II. 1. მეხსიერების გვერდობრივად განაწილება

თითოეული პროცესის ვირტუალური სამისამართო სივრცე დაიყოფა ფიქსირებული სიდიდის ნაწილებად, რომლებსაც ***ვირტუალური გვერდები*** ეწოდება. ზოგადად ვირტუალური სამისამართო სივრცის ზომა გვერდის ზომის ჯერადი არ არის, ამიტომ თითოეული პროცესის ბოლო გვერდს ემატება ფიქტური მიდამო.

კომპიუტერის ოპერატიული მეხსიერებაც იყოფა იმავე ზომის ნაწილებად, რომლებსაც ***ფიზიკური გვერდები*** (ანუ ***ბლოკები***) ეწოდება.

გვერდის ზომად ჩვეულებრივ *2*-ის ხარისხის ტოლი სიდიდე (მაგ., *512*, *1024*, *2048* და ა. შ) აირჩევა, რაც გვერდების გარდაქმნის მექანიზმის გამარტივების საშუალებას იძლევა.

პროცესის ჩატვირთვისას მისი ვირტუალური გვერდების ერთი ნაწილი თავსდება ოპერატიულ მეხსიერებაში, დანარჩენი ნაწილი კი დისკზე. სავალდებულო არაა მომიჯნავე ფიზიკურ გვერდებში მომიჯნავე ვირტუალური გვერდების განთავსება. ჩატვირთვისას ოპერაციული სისტემა თითოეული პროცესისათვის ქმნის საინფორმაციო სტრუქტურას – ***გვერდების ცხრილს***, რომელშიც მყარდება შესაბამისობა ოპერატიულ მეხსიერებაში ჩატვირთული ფიზიკური გვერდებისა და ვირტუალური გვერდების ნომრებს შორის, ან შეინიშნება იმის შესახებ, რომ ვირტუალური გვერდი გადატვირთულია დისკზე. გარდა ამისა გვერდების ცხრილი შეიცავს ისეთ მმართველ ინფორმაციებს, როგორიცაა გვერდის მოდიფიცირების ნიშანი, გადმოუტვირთაობის ნიშანი (ზოგიერთი გვერდების გადმოტვირთვა შეიძლება აკრძალული იყოს), გვერდთან მიმართვის ნიშანი (თვლისათვის გამოიყენება დროის გარკვეული პერიოდის განმავლობაში მიმართვების რაოდენობა) და ვირტუალური მეხსიერების მექანიზმის მიერ ფორმირებადი და გამოყენებადი სხვა მონაცემი.

მორიგი პროცესის გააქტიურებისას პროცესორის სპეციალურ რეგისტრში ჩაიტვირთება მოცემული პროცესის გვერდების ცხრილი.

მეხსიერებასთან ყოველი მიმართვისას გვერდების ცხრილიდან წაიკითხება ინფორმაცია იმ ვირტუალური გვერდის შესახებ, რომელზეც მოხდა მიმართვა. მოცემული ვირტუალური გვერდის ოპერატიულ მეხსიერებაში განთავსებისას ვირტუალური მისამართები გარდაიქმნება ფიზიკურ მისამართებად. მოცემულ მომენტში ვირტუალური

გვერდი დისკზე თუ არის გადატვირთული, მაშინ ხდება ე. წ. ***გვერდობრივი წყვეტა***: შესრულებადი პროცესი გადაიყვანება ლოდინის რეჟიმში და გააქტიურდება მზა პროცესების რიგში მდგარი სხვა პროცესი. პარალელურად გვერდობრივი წყვეტის დამამუშავებელი პროგრამა დისკზე პოულობს მოთხოვნილ ვირტუალურ გვერდს და ცდილობს იგი ჩატვირთოს ოპერატიულ მეხსიერებაში. მეხსიერებაში თავისუფალი ფიზიკური გვერდის არსებობისას ჩატვირთვა დაუყოვნებლივ მოხდება, საწინააღმდეგო შემთხვევაში წყდება საკითხი იმის შესახებ, თუ რომელი გვერდი შეიძლება გადაიტვირთოს მეხსიერებიდან დისკზე. ასეთი გვერდი შეიძლება მრავალი სხვადასხვა კრიტერიუმით შეირჩეს: შეიძლება გადაიტვირთოს გვერდი, რომელიც დიდი ხანია არ იყო გამოყენებილი, ან ნებისმერი გვერდი, ის გვერდი, რომელზეც ბოლო დროს მიმართვების რაოდენობა იყო მინიმალური.

ზოგიერთ სისტემაში გამოიყენება გვერდების მუშა სიმრავლის ცნება. ***გვერდების მუშა სიმრავლე*** ეწოდება ყველაზე ხშირად გამოყენებადი გვერდების ჩამონათვლისაგან წარმოქმნილ სიმრავლეს და იგი ფორმირდება თითოეული პროცესისათვის. ამ სიმრავლეში შემავალი გვედები მუდმივად უნდა იყოს ოპერატიულ მეხსიერებაში და ამიტომ მათი გადატვირთვა საჭირო არე არის.

ოპერატიული მეხსიერებიდან გასაძევებელი გვერდის ამორჩევის შემდეგ გაანალიზდება გვერდების ცხრილში ამ გვერდის შესახებ არსებული მოდიფიცირების ნიშანი. გასაძევებელი გვერდი თუ ჩატვირთვის შემდეგ იყო მოდიფიცირებული, მაშინ ოპერატიული მეხსიერებიდან მის გაძევებამდე დისკზე გადაიტვირთება მისი მოდიფიცირებული ვერსია, საწინააღმდეგო შემთხვევაში იგი შეიძლება უბრალოდ განადგურდეს და თავისუფლად გამოცხადდეს შესაბამისი ფიზიკური გვერდი.

განვიხილოთ გვერდობრივად ორგანიზებული მეხსიერებისას ***ფიზიკურ გვერდებად ვირტუალური გვერდების გარდაქმნის მექანიზმი.***

გვერდობრივი განაწილებისას ვირტუალური მეხსიერება შეიძლება (p,s) წყვილის სახით იქნეს წარმოდგენილი, სადაც p – პროცესის ვირტუალური გვერდის ნომერია (დანომვრა იწყება 0-დან), ხოლო s – ***წანაცვლება*** ვირტუალური გვერდის ფარგლებში. ვინაიდან გვერდის ზომაა 2^k, ამიტომ s წანაცვლება მიიღება ვირტუალური მისამართის kრაოდენობის უმცროსი თანრიგების უბრალო გამოცალკე-

ვებით. დარჩენილი უფროსი თანრიგების ერთობლიობა წარმოადგენს ვირტუალური გვერდის ***p*** ნომრის ორობით ჩანაწერს.

ოპერატიულ მეხსიერებასთან ყოველი მიმართვისას აპარატურული საშუალებები ასრულებს შემდეგ ქმედებებს:

- გვერდების ცხრილის საწყისი მისამართის *(გვერდების ცხრილის მისამართის რეგისტრის შიგთავსის)*, ვირტუალური გვერდის ნომრის (*ვირტუალური მისამართის უფროსი თანრიგების*) და გვერდების ცხრილში არსებული ჩანაწერის სიგრძის (*სისტემური კონსტანტას*) საფუძველზე განსაზღვრავს ცხრილში საჭირო ჩანაწერის მისამართი;
- ამ ჩანაწერიდან ამოიტანს ფიზიკური გვერდის ნომერს;
- ფიზიკური გვერდის ნომერს მიუერთებს წანაცვლებას (*ვირტუალური მისამართის უმცროს თანრიგებს*).

გვერდის ზომის ***2***-ის ხარისხის ტოლობის გამო შეკრების ხანგრძლივი ოპერაციის ნაცვლად შეგვიძლია გამოვიყენოთ ***კონკატენაციის*** (მიერთების) ოპერაცია, რაც ამცირებს ფიზიკური გვერდის მიღების დროს, ე. ი. ზრდის კომპიუტერის მწარმოებლურობას.

გვერდობრივად ორგანიზებული მეხსიერებიანი სისტემის მწარმოებლურობაზე გავლენას ახდენს გვერდობრივ შეწყვეტათა დამუშავებასა და ფიზიკურ მეხსიერებად ვირტუალური მეხსიერების გარდაქმნასთან დაკავშირებული დროითი დანახარჯები გვერდობრივი შეწყვეტების ხშირად წარმოშობისას სისტემას გვერდების სვოპირებისათვის დიდი დრო დაეხარჯება. გვერდობრივი შეწყვეტების სიხშირის შესამცირებლად, ***ერთი შეხედვით,*** კარგი იქნებოდა გაგვეზარდა გვერდების ზომა, რაც გვერდების ცხრილის ზომას, და, მაშასადამე, მეხსიერების დანახარჯებსაც, შეამცირებდა. მაგრამ დიდი გვერდების შემთხვევაში თითოეული პროგრამის ბოლო ვირტუალურ გვერდზე დიდი გამოუყენებელი თავისუფალი ადგილი დარჩება. საშუალოდ თითოეულ პროგრამაზე იკარგება გვერდის ზომის ნახევრი, რაც დიდი გვერდის დროს მნიშვნელოვანი სიდიდეა. ფიზიკურ გვერდად ვირტუალური გვერდის გარდაქმნის დროის სიდიდე მნიშვნელოვნადაა დამოკიდებული გვერდების ცხრილთან შეღწევის დროზე. ამიტომ სასურველია აღნიშნული ცხრილის განთავსებისათვის გამოვიყენოთ „სწრაფი“ დამხსომებელი მოწყობილობები (*სპეციალური რეგისტრების ნაკრები, ასოციური ძიების ან დისკების კეშირების გამომყენებელი მეხსიერება*).

II. 2. მეხსიერების სეგმენტურად განაწილება

მეხსიერების გვერდობრივად ორგანიზებისას პროცესორის სამისამართო სივრცე სხვადასხვა ნაწილებად ***მექანიკურად იყოფა***. ეს საშუალებას არ გვაძლევს, პროგრამის სხვადასხვა ნაწილებთან (სეგმენტებთან) შეღწევის ხერხის დიფერენცირება მოვახდინოთ. ეს უკანასკნელი კი ხშირად შეიძლება ძალიან სასურველი იყოს. მაგალითად, იგი საშუალებას მოგვცემდა ***კოდურ სეგმენტში*** აგვეკრძალა ჩაწერისა და წაკითხვის ოპერაციები, ხოლო ***მონაცემების სეგმენტში*** – მხოლოდ წაკითხვის ოპერაციის შესრულება გაგვეხადა შესაძლებელი. გარდა ამისა, ნაწილებად პროგრამის არა მექანიკური, არამედ „გააზრებული“ დაყოფა საშუალებას მოგვცემდა ***ერთი სეგმენტი*** რამდენიმე პროცესისათვის გამოგვეყენებინა. მაგალითად, ორი პროცესის მიერ ერთი და იგივე მათემატიკური ქვეპროგრამის გამოყენებისას ოპერატიულ მეხსიერებაში შეგვეძლო ამ ქვეპროგრამის მხოლოდ ერთი ასლი ჩაგვეტვირთა. არც ამის საშუალებას გვაძლევს სამისამართო სივრცის ზემოთ აღნიშნული მექანიკური დაყოფა.

განვიხილოთ, მეხსიერების სეგმენტური დანაწილებისას როგორ რეალიზდება აღნიშნული შესაძლებლობები. დამპროგრამებელი სეგმენტებში შესანახი ინფორმაციის აზრობრივი მნიშვნელობის შესაბამისად განსაზღვრავს სეგმენტების ზომებს და ამ ზომის სეგმენტებად დაყოფს პროცესის ვირტუალურ სამისამართო სივრცეს. ცალკეული სეგმენტი შეიძლება გამოეყოს ქვეპროგრამას, მონაცემთა მასივს და ა.შ. პროგრამა სეგმენტებად უსიტყვო შეთანხმების ძალით შეიძლება კომპილატორმაც დაყოს.

პროცესის ჩატვირთვისას სეგმენტების ნაწილი თავსდება ოპერატიულ მეხსიერებაში (თითოეული ამ სეგმენტისათვის თავისუფალი მეხსიერების შესაბამის ადგილს მოიძიებს ოპერაციული სისტემა), ნაწილი კი – დისკურ მეხსიერებაში. ერთი პროგრამის სეგმენტებმა შეიძლება დაიკავოს ოპერატიული მეხსიერების არამომიჯნავე უბნები. ჩატვირთვისას სისტემა (გვერდების ცხრილის ანალოგურად) ქმნის პროცესის სეგმენტების ცხრილს, რომელშიც თითოეული სეგმენტისათვის მითითებულია ოპერატიულ მეხსიერებაში სეგმენტის საწყისი ფიზიკური მისამართი, სეგმენტის ზომა, სეგმენტთან შეღწევის წესი, მოდიფიცირების ნიშანი, დროის ბოლო ინტერვალის განმავლობაში მოცემულ სეგმენტთან მიმართვის რაოდენობის მაჩვენებე-

ლი ნიშანი და სხვა ინფორმაცია. რამდენიმე პროცესის ვირტუალური სამისამართო სივრცე თუ მოიცავს ერთსა და იმავე სეგმენტს, მაშინ ამ პროცესების სეგმენტთა ცხრილებში შეიძლება არსებობდეს იმ ოპერატიული მეხსიერების ერთსა და იმავე უბნისათვის მიმართვები, რომელშიც სეგმენტის ერთადერთი ეგზემპლიარია ჩატვირთული.

სეგმენტურად ორგანიზებული სისტემა ზუსტად გვერდობრივად ორგანიზებული სისტემის ანალოგურად ფუნქციონირებს: დროდადრო ხდება მეხსიერებაში საჭირო სეგმენტების არარსებობით გამოწვეული შეწყვეტები, მეხსიერების გათავისუფლების საჭიროებისას ზოგიერთი სეგმენტი დისკზე გადაიტვირთება. ოპერატიულ მეხსიერებასთან ყოველი მიმართვის დროს ფიზიკურ მისამართებად გარდაიქმნება ვირტუალური მისამართები. გარდა ამისა, მეხსიერებასთან მიმართვისას მოწმდება მოცემულ სეგმენტთან მოცემული ტიპის შეღწევის დასაშვებულობა.

მეხსიერების სეგმენტურად ორგანიზებისას ვირტუალური მეხსიერება წარმოიდგინება (***g, s***) წყვილის სახით, სადაც ***g*** არის სეგმენტის ნომერი, ხოლო ***s*** – წანაცვლება სეგმენტში. ფიზიკური მიასმართი მიიღება სეგმენტის ***g*** ნომრის მიხედვით სეგმენტების ცხრილში ნაპოვნი სეგმენტის საწყის ფიზიკურ მისამართისათვის ***s*** წანაცვლების მიმატებით.

მეხსიერების სეგმენტურად ორგანიზების მეთოდის ნაკლია სეგმენტების დონეზე ფრაგმენტირების წარმოშობა და გვერდობრივ ორგანიზების მეთოდთან შედარებით მისამართის უფრო ნელი გარდაქმნა.

II. 3. მეხსიერების გვერდობრივ-სეგმენტური განაწილება

მოცემულ შემთხვევაში ხდება მეხსიერების გვერდობრივად და სეგმენტურად განაწილების კომბინირება, რაც ორივე მიდგომისათვის დამახასიათებელი ღირსებების შეხამების საშუალებას იძლევა.პროცესის ვირტუალური სივრცე იყოფა სეგმენტებად, ხოლო თითოეული სეგმენტი – გვერდებად. გვერდები ინომრება სეგმენტის ფარგლებში. ოპერატიული მეხსიერება დაიყოფა ფიზიკურ გვერდებად. პროცესს ოპერაციული სისტემა ჩატვირთავს გვერდების სახით, რომლის დროსაც გვერდების ნაწილი განთავსდება ოპერატიულ მეხსიერებაში, ნაწილი კი დისკზე. თითოეული პროცესისათვის იქმნება სეგმენტების ცხრილი, რომელშიც ამ პროცესის ყველა სეგმენ-

ტისათვის მითითებულია გვერდების ცხრილთა მისამართები. სეგმენტების ცხრილის მისამართი შესაბამისი პროცესის გააქტიურები-სას პროცესორის სპეციალურ რეგისტრში ჩაიტვირთება.

II. 4. სვოპინგი (გადაცვლა).

სვოპინგი ვირტუალური მეხსიერების ნაირსახეობაა. კვლევებით დადგენილია, რომ პროცესორის ***90%***-ით დატვირთვისათვის საკმარისია მასში სულ სამი პროცესი მიმდინარეობდეს. სხვაგვარადაა საქმე ***ინტერაქტიული ამოცანების*** დროს. პროცესორის იმავე დატვირთვით მუშაობისათვის საჭიროა სისტემაში იყოს არა სამი, არამედ ათობით ასეთი ამოცანა. ამოცანის შესასრულებლად აუცილებელია იგი ჩაიტვირთოს ოპერატიულ მეხსიერებაში, რომლის მოცულობა შეზღუდულია. ასეთ პირობებში გამოთვლითი პროცესის ორგანიზებისათვის შემოთავაზებული იქნა ***სვოპინგად*** წოდებული მეთოდი. ამ მეთოდის შესაბამისად ზოგიერთი (ჩვეულებრივ, მომლოდინე) ამოცანა დროებით ***მთლიანად*** გადაიტვირთება დისკზე. მიუხედავად იმისა, რომ ეს ამოცანები დისკზეა და არა ოპერატიულ მეხსიერებაში, ისინი მაინც რჩება ***ოპერაციული სისტემის დამგეგმავის*** მხედველობის არეში: რომელიმე მათგანის გააქტიურების მომენტის დადგომისთანავე იგი დისკიდან ოპერატიულ მეხსიერებაში ***მთლიანად*** გადაიტანება.ამისათვის საჭირო ზომის თავისუფალი ადგილის მეხსიერებაში არ არსებობისას ოპერატიულ მეხსიერებაში გადაიტანება სხვა ამოცანა. ვირტუალური მეხსიერების რეალიზების ადრე განხილული მეთოდებისაგან განსხვავებით მოცემული მეთოდის თავისებურებაა ის, რომ **<u>ამოცანა არ ნაწილდება ოპერატიულ და დისკურ მეხსიერებებს შორის: იგი მთლიანადაა მოთავსებული ან ერთ, ან მეორე მათგანში.</u>** არსებობს დისკურიმეხსიერებიდან ოპერატიულ მეხსიერებაში ჩასატვირთად, ან პირიქით, გადასატვირთავად კონკრეტული ამოცანის შერჩევის სხვადასხვა ხერხი.

II თავი
ფაილები და ფაილური სტრუქტურები

ფაილური სისტემების შესახებ

1. ზოგადი ცნობები

ყოველდღიური ყოფითი ამოცანების გადასაწყვეტად ჩვენ ხშირად ვიყენებთ როგორც გარკვეული სახის კომპიუტერული ტექნიკას (დესკტოპებს, ლეპტოპებს, სმარტფონებს და ა.შ), ასევე კომპიუტერულ პროგრამებსაც. ორივე მათანს ***კომპიუტერული გამოყენებები*** („*computer applications*“) ანუ უბრალოდ ***გამოყენებები*** ეწოდება. დიფერენცირების მიზნით პირველს აპარატურულ, ხოლო მეორეს – პროგრამულ გამოყენებებს უწოდებენ. შემდგომში ჩვენ მხოლოდ პროგრამულ გამოყენებებთან გვექნება საქმე, ამიტომ ტერმინ ***„გამოყენების“*** ხმარებისას სწორედ ისინი გვექნება მხედველობაში.

კომპიუტერული გამოყენებები შეინახება სპეციალურ საცავში, საიდანაც საჭიროებისამებრ შეიძლება მათი გამომძახება. ასეთ საცავად ვირტუალური სამისამართო სივრცის გამოყენებას აქვს შემდეგი სამი ნაკლი: ***1)არ შეუძლია დიდი ზომის გამოყენებების შენახვა.*** საცავის ტევადობას ზღუდავს ვირტუალური სამისამართო სივრცის ზომა. ზოგიერთი გამოყენებისათვის ეს ზომა სრულიად საკმარისია, მაგრამ მასში ვერ დაეტევა ისეთი გამოყენებები, როგორიცაა სარკინიგზო და ავიაბილეთების დარეზერვების სისტემა, საბანკო აღრიცხვის სისტემა და ა.შ.; ***2)გამოყენებას ვერ ინარჩუნებს მუდმივად.*** კომპიუტერული გამოყენებები ხანგრძლივად უნდა იქნეს შენახული, ხოლო პროცესის დამთავრების შემდეგ მისი ვირტუალური სამისამართო სივრცე ქრება და იკარგება ამ მასში შენახული გამოყენება; ***3) ვერ უზრუნველყოფს გამოყენებასთან რამდენიმე პროცესის შეღწევას.***კომპიუტერულ გამოყენებაში არსებულ გამომოყენებასთან, ან მის ნაწილთან, ხშირად რამდენიმე პროცესს სჭირდება შეღწევა. ისეთ

გამოყენებას კი, როგორიცაა ინტერაქტიული სატელეფონო ცნობარი, ერთი პროცესის ვირტუალურ სამისამართო სივრცეში თუ შევინახავთ, მასთან შეღწევის უფლება მხოლოდ ამ პროცესს ექნება.

ზემოთ აღნიშნულის გამო გამოყენებების ხანგრძლივად შესანახად დიდი ხნის განმავლობაში გამოიყენებოდა დისკური მეხსიერება. სამწუხაროდ, მისი მწარმოებლურეობა ძალიან დაბალია. გარდა ამისა, დისკი შეიძლება წარმოვიდგინოთ ფიქსირებული ზომის ბლოკების წრფივი მიმდევრობის სახით, რომელთანაც ურთიერთობისათვის საკმაოდ ბევრი ოპერაციის შესრულება გვიხდება, რომელთაგანაც შეიძლება გამოვყოთ სასურველ ***k*** ბლოკში არსებული ინფორმაციის წაკითხვისა და ამ ბლოკში ახალი ინფორმაციის ჩაწერის პროცედურები. ამ ორი პროცედურის შესრულება, სხვა ოპერაციებზე რომ არაფერი ვთქვათ, ძალიან მოუხერხებელია (განსაკუთრებით ბევრი გამოყენების შემცველი დიდი სისტემების დროს, ან რამდენიმე მომხმარებლის არსებობისას). ამ დროს წამოიჭრება მრავალი პრობლემა, რომლებიც დაკავშირებულია თავისუფალი ბლოკის დადგენასა და საჭირო ინფორმაციის მოძიებასთან, ერთი მომხმარებლის ინფორმაციის სხვა მომხმარებლისაგან დაცვასთან და ა. შ.

როგორც წინა თავში დავინახეთ, ***პროცესის ცნების*** უნივერსალური გამოყენებისათვის ფიზიკური მეხსიერების ნაცვლად შემოღებულია აბსტრაქტული ვირტუალური სივრცის ცნება. სხვა სიკეთეებთან ერთად ამან დამპროგრამებლის შრომაც გაამარტივა: პროგრამის დაწერისას იგი ყურადღებას არ აქცევს ოპერატიული მეხსიერების რეალურ სიდიდეს და პროგრამას წერს არარსებული ვირტუალური მეხსიერებისათვის. ზემოთ ჩამოთვლილი პრობლემების გადასაწყვეტად და კომპიუტერული გამოყენებებთან მომხმარებელთა ურთიერთობის გამარტივებისათვის აუცილებელი ხდება კიდევ ერთი აბსტრაქციის – ***ფაილის*** შემოღება. პროცესი (ნაკადი), ვირტუალური სამისამართო სივრცე და ფაილი ისეთი უმნიშვნელოვანესი აბსტრაქციებია, რომელთა არსის ცოდნა აუცილებელია ოპერაციული სისტემის ფუნქციონირების ნათლად გააზრებისათვის.

ფაილები პროცესების მიერ შექმნილი ლოგიკური საინფორმაციო ბლოკებია. დისკზე ჩვეულებრივ ათასობითი და მილიონობითი რაოდენობის ურთიერთდამოუკიდებელი ფაილია შენახული. სამისამართო სივრცის სახესხვაობად თითოეული ფაილის განხილვა ჭეშმარიტება-

სთან ახლო იქნება, ოღონდ უნდა გვახსოვდეს, რომ ***ფაილები გამოიყენება არა ოპერატიული მეხსიერების, არამედ დისკური მეხსიერების მოდელირებისთვის.***

პროცესებს შეუძლია წაიკითხოს არსებული და, საჭიროებისამებრ შექმნას ახალი ფაილები. ფაილებში ინფორმაცია შეინახება ხანგრძლივად ფაილი არსებობას მაშინ შეწყვეტს, თუ მას გააძევებს მისი მფლობელი.

ფაილის ცნებასთან განუხრელადაა დაკავშირებული ფაილური სისტემის ცნებაც. ***ფაილური სისტემა*** ოპერაციული სისტემის ნაწილია, რომელსაც ევალება შემდეგი ორი ფუნქციის შესრულება: ***1)*** მომხმარებელს შეუქმნას დისკზე შენახულ მონაცემებთან სამუშაოდ მოსახერხებელი ინტერფეისი ***2)***რამდენიმე მომხმარებელსა დაპროცესს მისცეს ფაილების ერთობლივი გამოყენების საშუალება.

ფაილური სისტემის ცნების ქვეშ ფართო გაგებით მოიაზრება: ***1)*** დისკზე არსებულ ფაილთა მთელი ერთობლიობა; ***2)*** მონაცემების სტრუქტურათა ნაკრებები, რომლებიც გამოყენებულია ფაილების მართვისათვის; ასეთი ნაკრებებია ფაილებისაგან შედგენილი კატალოგები, ფაილთა დესკრიპტორები, დისკზე თავისუფალ და დაკავებულ სივრცეთა განაწილების ცხრილები; ***3)*** სისტემურ პროგრამულ საშუალებათა კომპლექსი, რომლებიც ასრულებს ფაილების მართვის ისეთ ფუნქციებს, როგორიცაა, მაგალითად, ფაილის შექმნა და განადგურება, ფაილისწაკითხვა ან მასში ინფორმაციის ჩაწერა, ფაილის დასათაურება, მოძებნა და ა. შ.

მოკლედ განვიხილოთ ფაილთან და ფაილურ სისტემასთან დაკავშირებული ძირითადი საკითხები.

2 ფაილების სახელები

ფაილები იდენტიფიცირდება საკუთარი სახელებით. ეს სპეციფიკური თვისება ფაილის ცნების განსაზღვრებისათვისაც შეიძლება გამოვიყენოთ და ვამტკიცოთ, რომ ***ფაილი*** ეწოდება მონაცემების ისეთ ერთობლიობას, რომელშიც შეღწევა მისი სახელითაა შესაძლებელი. მაშასადამე, ფაილთან ურთიერთობისათვის (ინფორმაციის წასაკითხად ან ჩასაწერად) მისამართის ნაცვლად მისი სახელის ცოდნაა საკმარისი.

ფაილის სახელი შედგება წერტილით გაყოფილი ორი ნაწილისაგან. წერტილის მარჯვნივ არსებული ნაწილი ***არასავალდებულ ნაწილად*** ითვლება და გამოიყენება მოცემული ფაილების შესახებ დამა-

ტებითი ინფორმაციის მისაღებად. ამიტომ მას ***სახელის გაფართოებასაც*** (***filenameextension***)უწოდებენ. წერტილის მარცხნივ არსებული ნაწილი ფაილის სახელის ***სავალდებულო ნაწილია***, რომელსაცხშირად ფაილის სახელადაც თვლიან. აღნიშნულიდან გამომდინარე, დასაწყისში სწორედ მას განვიხილავთ.

მომხმარებელი სიმბოლურ სახელსქმნისდა ფაილსმასარქმევს (ანიჭებს). სახელის შექმნისას მომხმარებელმა უნდა დაიცვას ოპერაციული სისტემის მიერ ფორმირებული მოთხოვნები, რომლითაც განისაზღვრება სახელის შესაქმნელად გამოყენებული სიმბოლები და სახელის სიგრძე. ოპერაციული სისტემა ბოლო პერიოდამდე მოითხოვდა მოკლე ყოფილიყო ფაილის სახელი. მაგალითად, ***ბილ გეიტსისა*** და ***მაკ მაკდონალდის*** მიერ ***1976-77*** წლებში დამუშავებული კლასიკური ფაილური ***FAT*** სისტემა(***File Allocation Table*** – „ფაილების განთავსების ცხრილი“), რომელიც წარმოადგენდა ძირითად ფაილურ სისტემას ***DOS*** და ***Windows*** ოჯახის ოპერაციულ სისტემებში (გარდა ***Windows NT***–ს ოჯახისა) და რომელიც სიმარტივის გამო დღესაც გამოიყენება ფლეშდამგროვებლებში, ფაილის სიგრძე ***8-3*** პროპორციით იზღუდება (*რვა სიმბოლო – სახელისათვის, სამი სიმბოლო – სახელის გასაფართოებლადგამოიყენება*). ***1983*** წელს გამოსული მეხუთე სერიის ოპერაციული ***UNIX*** სისტემის მოთხოვნით ფაილის სახელი შეიძლება მხოლოდ ***14***-მდე სიმბოლოს შეიცავდეს.

მომხმარებლისათვის გაცილებით მოსახერხებელია ფაილების სახელებისათვის გამოიყენოს გრძელი სახელები, რაც მას ***მნემომონიკური*** (*ანუ ადვილად დამახსოვრებადი*) ***სახელის*** შექმნის საშუალებას აძლევს (*ძვ. ბერძ.„**μνημονικόν**“ – „დამახსოვრების ხელოვნება“*). მნემონიკური სახელით მომხმარებელს ადვილად შეუძლია გაიხსენოს დიდი ხნის წინათ მის მიერ შექმნილი ფაილების შინაარსი.

თანამედროვე ფაილური სისტემები, როგორც წესი, ფაილებისათვის გრძელი სახელების დარქმევის საშუალებას გვაძლევს. მაგალითად, ოპერაციული სისტემა ***Windows NT***–ის ოჯახის წევრებში გამოყენებული ფაილური ***NTFS*** სისტემის თანახმად ფაილის სახელი შეიძლება შედგებოდეს **255** სიმბოლოსაგან, რომელსაც ემატება დამამთავრებელი ნულოვანი სიმბოლო.

გრძელ სახელებზე გადასვლისას წარმოიქმნება ადრე შექმნილ მოკლე სახელებიან ფაილებთან გრძელი სახელებიანი ფაილების შე-

თავსებადობის პრობლემა. ადრეული შეთანხმებების დაცვით ფაილებთან მიმართვისათვის ***გამოყენებებს*** უნდა შეეძლოს ფაილების გრძელი სახელები წარმოადგინოს ეკვივალენტური მოკლე სახელების (ფსევდონიმების) სახით.

გრძელ სახელებს ცნობილი ძველი ფაილური სისტემების ახალი ვერსიებიც იყენებს. მაგალითად, ოპერაციულ სისტემა ***Windows 95***-ში არსებული ფაილური ***VFAT*** სისტემა ფაილებისათვის გრძელი სახელის დარქმევის საშუალებასაც იძლევა. გარდა ამისა, იგი არ ცვლიდა დისკზე არსებული მონაცემების სტრუქტურას.

ჩვეულებრივ სხვადასხვა ფაილს შეიძლება ჰქონდეს ერთნაირი სიმბოლური სახელი. ამ შემთხვევაში ფაილი იდენტიფიცირდება ე. წ. ***შედგენილი სახელით***, რომელიც წარმოადგენს კატალოგთა სიმბოლური სახელების მიმდევრობს. ზოგიერთ სისტემაში ერთსა და იმავე ფაილს არ შეიძლებოდა დარქმეოდა სხვადასხვა სახელი, მაგრამ არსებობს ამ შეზღუდვის არმქონე სისტემებიც. უკანასკნელ შემთხვევაში ოპერაციული სისტემა ფაილს ანიჭებს ისეთ დამატებით ***უნიკალურ სახელს***, რომ შეიძლება დამყარდეს ურთიერთცალსახა დამოკიდებულება ფაილსა და მის უნიკალურ სახელს შორის. უნიკალური სახელი წარმოადგენს რიცხვით იდენტიფიკატორს და მას გამოიყენებს ოპერაციული სისტემის პროგრამები. ფაილის ასეთი უნიკალური სახელის მაგალითია***UNIX*** სისტემაში ინდექსური დესკრიპტორის ნომერი. (*ლათ.* ***დესკრიპტორი;*** *ქართ.: ქართულად ნიშნავს „აღმწერს“. გამოიყენება დოკუმენტის ძირითადი შინაარსის შინაარსის აღწერისათვის.* ***ინდექსური დესკრიპტორში*** *ფიქსირებულია ფაილის ტიპი, მასთან შეღწევის წესი, ზომა და ა. შ. თითოეულ ფაილს დისკზე შეესაბამება საკუთარი რიგითი ნომრით იდენტიფიცირებადი ერთადერთი ინდექსური დესკრიპტორი*).

ახლა რამდენიმე სიტყვით შევეხოთ ფაილის სახელის არასავალდებულო ნაწილს – ***სახელის გაფართოებას***. ფაილის სახელის გაფართოება ახდენს ***ფაილის ტიპის (ფორმატის) იდენტიფიცირებას და მიუთითებს,*** რომელი პროგრამები იყენებს ამ ფაილებს. იგი წარმოადგენს ფაილის სახელისათვის დამატებული სიმბოლოების ერთობლიობას. დაინტერესებულ პირებს მათ შესახებ ყველა საჭირო ინფორმაცია შეუძლიათ ***2.1*** ცხრილიდან მიიღოს.

ცხრილი 2.1. ფაილის ტიპები (დასაწყისი)

გაფართოების სახე	ფაილის ტიპი (ფორმატი)	პროგრამა
•exe	შემსრულებელი ფაილი – შესასრულებლად გამზადებული პროგრამების შემსრულებელი ფაილი	***Windows***-ის, ***DOS***-ის,***Simbian***-ის, ***OS/2***-ს ნებისმიერი სამუშაო პროგრამა
•msi	პროგრამების მაინსტალირებელი – პროგრამების დამყენებელი ფაილები	დასაყენებელ პროგრამათა პაკეტები
•doc ან •docx	დოკუმენტი ***Word 2007***-ის ზედა ტექსტურ რედაქტორებში	ტექსტური პროცესორი ***Word***
•xls ან •xlsx	ცხრილების ფაილი ***Excel 2007***-ის ზედა ცხრილურ რედაქტორებში	ცხრილური პროცესორი ***Exel***
•txt	მარტივი ფორმატის ტექსტური ფაილი (დოკუმენტი)	ტექსტური რედაქტორის ბლოკნოტი
•ppt ან •pptx	პრეზენტაცია Power-Point-ის ფაილი	საპრეზენტაციო პროგრამა ***Power- Point***
•accdb	მონაცემთა ბაზა ***Access***-ის ფაილი	სისტემა ***Access***
•fla, •mp3, •are, •waw, •acc, •ac3, •m4a, •ogg, •wma, და ა.შ.	ბგერითი (ციფრული) ფაილი	ნებისმიერი აუდიო პლეერი
•bmp, •jpg, (jpg)j, •pnj,•gif, •tiff, •ico,•raw.	გამოსახულების ფაილი	გამოსახულების სტანდარტული მენეჯერები; სპეცპროგრამები
•avi, •wmw, •mkv, •3gp, •flv, •mpeg, •mp4, •mov , •vpb.	ვიდეოფაილები	სხვადასხვა პლეერში
•djvu	შეკუმშულ გამოსახულებათა ფაილი	djvu ფაილების ნებისმიერ წამკითხავი პროგრამაში
•swf, •flv	ინტერნეტის ფლეშ ან ვიდეოფაილები	აღიქვამს ფლეშფირსაკრავიანი ნებისმიერი ბრაუზერი
•pdf	ელექტრონული დოკუმენტის ფაილი	Adobe Reader და სხვებში

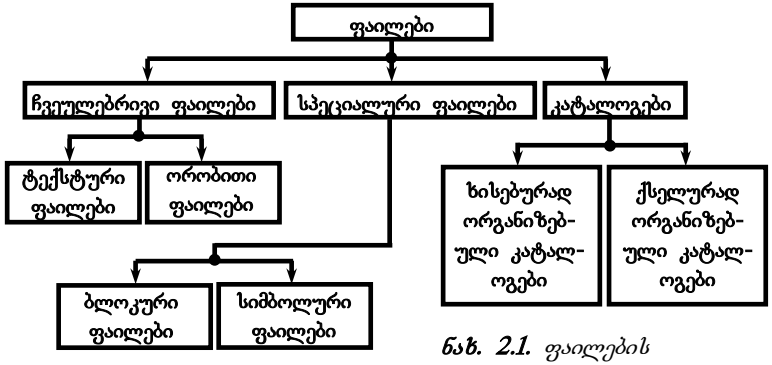
დასასრული ნახეთ მომდევნო გვერდზე

ცხრილი 2.1. ფაილის ტიპები (დასასრული)

გაფართოების სახე	ფაილის ტიპი (ფორმატი)	პროგრამა
•rar, •zip, 7z, •tar, •gzip, •gz, •jar	საარქივო კონტეინერის ფაილები	პოპულარული არქივებისათვის
•php. •htm •htm,	ვებგვერდის ფაილები	ბრაუზერებში

3. ფაილების სახეები

კლასიფიკაციის თანახმად (ნახ. *2.1*) ფაილები შეიძ-ება დავაჯგუფოთ ჩვეულებრივ ფაილებად, სპეციალურ ფაილებად და კატალოგებად ანუ დირექტორიებად. მოკლედ განვიხილოთ თითოეული მათგანი.



ნახ. 2.1. ფაილების

3.1. ჩვეულებრივი ფაილები

ჩვეულებრივი ფაილები ეწოდება მომხმარებლის ინფორმაციის შემცველ ფაილებს. ერთმანეთისაგან განასხვავებენ ტექსტურ და ორობით ფაილებს.

ტექსტური ფაილები წარმოადგენს ***ASCII*** კოდით გამოსახული სიმბოლოებისაგან შედგენილ სტრიქონების ერთობლიობას. თანამედმეროვე სისტემებში სტრიქონები დაყოფილია სტრიქონების დამყოფებით (სტრიქონებს შორის არსებული წყვეტებით). ადრეულ სისტემებში სტრიქონები შეინახებოდა მუდმივი ან ცვლადი სიდიდის ჩანაწერების სახით. ზოგჯერ (განსაკუთრებით მაშინ, როდესაც ფაილურ სისტემაში არ შეინახება ინფორმაცია ფაილის ზომის შესახებ) ტექსტური ფაილის ბოლო აღინიშნება სპეციალური ნიშნით. მისი

ღირსებებია: უნივერსალურობა (მისი წაკითხვა შეუძლია ნებისმიერ სისტემას); მდგრადობა (მასში ბაიტების დაზიანებისას შეიძლება მონაცემების აღდგენა) და სიმარტივე, ხოლო ***ნაკლია*** არაეკონომიურება (იკავებს მინიმალურად აუცილებელზე მეტ ადგილს) და მათზე შესასრულებული ზოგიერთი ოპერაციის არაეფექტურობა (მაგალითად, მეათასე სტრიქონზე გადასასვლელად საჭიროა ***999*** სტრიქონის წაკითხვა).

ორობითი ფაილები არ იყენებს ***ASCII***კოდებს; იგი შედგება ორობითი ციფრებისაგან (ბიტებისაგან) შედგენილი ბაიტების ერთობლიობისაგან. მათი შინაგანი სტრუქტურა ხშირად რთულია.

3.2 სპეციალური ფაილები.

სპეციალური ფაილები ეწოდება შეტანა/გამოტანის მოწყობილობებთან ასოცირებულ ფაილებს. ისინი მომხმარებელს ფაილში ჩაწერისა და ფაილიდან წაკითხვის ჩვეულებრივი ბრძანებების გამოყენებით შეტანა/გამოტანის ოპერაციების შესრულების შესაძლებლობას აძლევს. ამ ბრძანებებს ჯერ დაამუშავებს ფაილური სისტემის ბრძანებები, ხოლო შემდეგ მოთხოვნის შესრულების გარკვეულ ეტაპზე ოპერაციული სისტემა მათ შესაბამისი მოწყობილობის მართვის ბრძანებად გარდაქმნის.

შეტანა/გამოტანის მოწყობილობები იყოფა ***ბლოკურად ორიენტირებულ*** და ***ბაიტურად ორიენტირებულ მოწყობილობებად***. ბლოკურად ორიენტირებული მოწყობილობები ინფორმაციას მქონე ბლოკებში, რომელთა დამისამართება შესაძლებელია (მათ აქვს საკუთარი მისამართები). აღნიშნულის გამო ბლოკურად ორიენტირებულ შეტანა/გამოტანის მოწყობილობებს დამისამართებად მოწყობილობებს უწოდებენ: შესაძლებელია ინიცირდეს მათი შემადგენელი ბლოკების ძიების პროცესი. ბაიტურად ორგანიზებული მოწყობილობები არადამისამართებადი მოწყობილობებია (არ შეიცავს დამისამართებულ ბლოკებს) და ამდენად ძიების პროცესის ინიცირების საშუალებას არ იძლევა: ისინი წარმოშობს ან მოიხმარს ბაიტებს. ბლოკურად ორგანიზებული მოწყობილობის კლასიკური მაგალითია დისკი, ხოლო ბაიტურად ორგანიზებული მოწყობილობის მაგალითებია ტერმინალები, პრინტერები და ქსელური ადაპტერები.

ზემოთ მოყვანილ შეტანა/გამოტანის ორ განსხვავებულ მოწყობილობებთან გამოსაყენებლად დამუშავდა ასევე ორი განსხვავებული სახის ფაილები. პირველს ***ბლოკური*** ანუ ***ბლოკურად ორიენტირებუ-***

ლი ფაილები, მეორეს კი – ***სიმბოლური*** ანუ ***ბაიტურად ორიენტირებული*** ფაილები ეწოდა (იხ. ***ნახ. 2.1***).

ბლოკური ფაილები ვერ იმუშავებს ბაიტურად ორიენტირებულ მოწყობილობასთან, მაგრამ სიმბოლურ ფაილებს შეუძლია ბლოკურად ორგანიზებულ ფაილებთან მუშაობაც. ეს იმითაა განპირობებული, რომ ბლოკურად ორიენტირებული მოწყობილობა (მაგალითად, დისკი) არამარტო ბლოკების, არამედ ბაიტების ნაკრებიცაა (პირველი ბაიტს იწყებს პირველი ბლოკი, ხოლო ბოლო ბაიტს დაასრულებს ბოლო ბლოკი). მოწყობილობის კონტროლერთან ფიზიკური გაცვლა ძველებურად ბლოკებად ხდება, მაგრამ მოწყობილობის სიმბოლური ფაილი ბლოკებს გარდაქმნის ბაიტების მიმდევრობა.

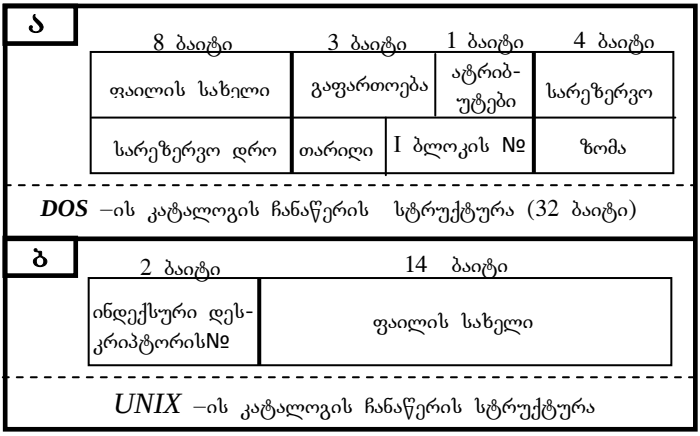
3. კატალოგები

კატალოგი, ერთი მხრივ, მომხმარებლის მიერ გარკვეული მოსაზრებით გაერთიანებული ფაილების ჯგუფია (მაგალითად, მასში შეიძლება გაერთიანდეს თამაშთა პროგრამების, ან ერთი პროგრამული პაკეტის შემცველი ფაილები), ხოლო მეორე მხრივ - ფაილებისა და მათი მდგენელების შესახებ სისტემური ინფორმაციის შემცველი ფაილიცაა. კატალოგში შედის მასში არსებული ფაილების სია, გარდა ამისა, მასში დგინდება შესაბამისობა ფაილებსა და მათ მახასიათებლებს (ატრიბუტებს) შორის.

ცხრილი 2.2 ფაილური სისტემების ატრიბუტებად გა*მოყენებული მახასიათებლები*

№	მახასიათებლები	№	მახასიათებლები
1	ინფორმაცია ნებადართული	10	ნიშანი «დროებითი»(გაძევდეს პროცესის დასრულების
2	ფაილთან შეღწევის	11.	ბლოკირების ნიშანი;
3	ფაილის მფლობელი;	12	ჩანაწერის
4	ფაილის შემქმნელი;	13	ჩანაწერში საგასაღებო ველის
5	ნიშანი «მხოლოდ წასაკითხად»	14	გასაღების
6	ნიშანი «ფარული ფაილი»	15	შექმნის, უკანასკნელი შეღწევის,
7	ნიშანი «სისტემური ფაილი»	16	ფაილის მიმდინარე
8	ნიშანი «საარქივო ფაილი»	17	ფაილის მაქსიმალური ზომა;
9	ნიშანი		

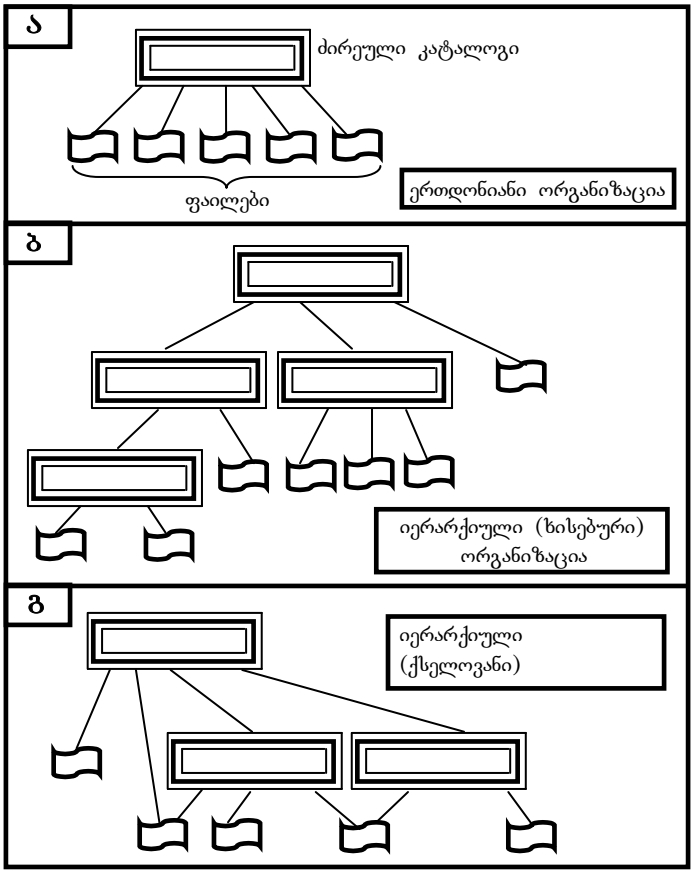
სხვადასხვა ფაილურ სისტემაში ატრიბუტებად გამოყენებული შესაძლო მახასიათებლები ***2.2.*** ცხრილშია მოყვანილი.



ნახ.2.2. *კატალოგების სტრუქტურა*

კატალოგები უშუალოდ შეიცავს ფაილების მახასიათებელთა მნიშვნელობებს, როგორც ეს გვაქვს ფაილურე ***MS-DOS*** სისტემაში (ნახ. ***2.2,ა***), ან ისინი მიუთითებს ამ მახასიათებლების შემცველ ცხრილებს, როგორც ესაა რეალიზებული ოპერაციულ ***UNIX*** სისტემაში (ნახ. ***2.2,ბ***). დაბალი დონის კატალოგები შეიძლება შევიდეს მაღალი დონის კატალოგებში, რაც იძლევა კატალოგებით იერარქიული სტრუქტურის წარმოქმნის შესაძლებლობას (ნახ. ***2.3***).

კატალოგების იერარქიით სხვადასხვაგვარი სტრუქტურა წარმოიშობა. ფაილი თუ მარტო ერთ კატალოგში შეიძლება შევიდეს, მაშინ მიიღება ***ხისებრი სტრუქტურა***, ხოლო თუ მას რამდენიმე კატალოგში შესვლის უფლება აქვს, წარმოიქმნება ქსელოვანი სტრუქტურა. ოპერაციულ სისტემა ***MS-DOS***-ში კატალოგები წარმოქმნის ხისებრ სტრუქტურებს (იხ. ნახ. ***2.3,ბ***), ხოლო ოპერაციულ სისტემა ***UNIX***-ში – ქსელოვან სტრუქტურას (იხ. ნახ.***2.3,გ***). კატალოგის, როგორც სპეციფიკური ფაილის, აქვს სიმბოლური სახელი და იგი ცალსახად იდენტიფიცირდება შედგენილი სახელით, რომელიც შეიცავს ყველა იმ კატალოგის სახელს, რომელიც არსებობს ძირეული კატალოგიდან მოცემულ კატალოგამდე არსებულ გზაზე.



ნახ. 2.3. *ფაილური სისტემის ლოგიკური ორგანიზაცია*

4. ფაილური

ფაილური სისტემაოპერაციული სისტემის ნაწილია, რომლის დანიშნულებაა, უზრუნველყოს დისკზე (გარე მეხსიერებაში) შენახულ მონაცემების ეფექტური გამოყენება და მომხმარებელს შეუქმნას ამისათვის მოსახერხებელი ინტერფეისი.

მაგნიტურ დისკზე ინფორმაციის შენახვის ორგანიზება არაა მარტივი. ამისათვის, მაგალითად, აუცილებელია კარგად ვიცოდეთ დისკის კონტროლერის ფუნქცონირების წვრილმანები და რეგისტრებთან მისი ურთიერთობის თავისებურებები. აპარატურასთან ურთიერთ-

ზემოქმედებისაგან მომხმარებლის დასაცავად შეიქმნა ფაილური სისტემის აბსტრაქტული მოდელი. მოწყობილობებთან მუშაობისათვის საჭირო დაბალი დონის ოპერაციებზე გაცილებით უფრო ადვილია ფაილებში ჩაწერისა და მისი წაკითხვის ოპერაციები.

გარე მეხსიერების (დისკის) გამოყენების ძირითადი იდეაა შემდეგი. ოპერაციული სისტემა მეხსიერებას ჰყოფს ფიქსირებული ზომის ბლოკებად, ვთქვათ ***4096*** რაოდენობის ბაიტებად. ფაილი ჩვეულებრივ წარმოადგენს ერთბაიტური ჩანაწერების არასტრუქტურირებულ მიმდევრობას, რომელიც დისკზე ბლოკების მიმდევრობების სახით შეინახება. ეს მიმდევრებები არაა აუცილებელი ერთმანეთის გვერდით იყოს განთავსებული. თითოეულ ბლოკში შენახული ჩანაწერების რაოდენობა მთელი რიცხვით გამოისახება.

ფაილების სისტემა საშუალებას გვაძლევს მეორეულ მეხსიერებაში არსებულ იმ ბლოკებს, რომლებშიც კონკრეტული ფაილია შენახული, ცნობარების (კატალოგების, დირექტორიების) დახმარებით „გადავაბათ“ ამ ფაილის უნიკალური სახელი. ეს საშუალებას გვაძლევს აღნიშნული სახელით (და არა მისამართის საშუალებით) დავამყაროთ ურთიერთობა ამ ბლოკებში ჩაწერილი ინფორმაციასთან, ანუ მათში შენახული ფაილთან. ეს აჩქარებს ინფორმაციის გამოყენების პროცესს. თავად ფაილებთან და მათი მართვისათვის გამოყენებულ მონაცემთა სტრუქტურებთან (კატალოგებთან, ფაილის დესკრიპტორთან, გარე მოწყობილობების განაწილების სხვადასხვა ცხრილთან) ერთად ***ფაილური სისტემა***მოიცავს ფაილებზე სხვადასხვა ოპერაციის მარეალიზებელ პროგრამულ საშუალებებსაც. ***ფაილური სისტემის ძირითადი ფუნქციებია: 1)*** ურთიერთობა მისთვის გამოყოფილ სივრცესთან; ***2***) გარე მეხსიერების განაწილება ფაილებს შორის; ***3***) საიმედოობისა და მტყუნებამდგრადობის დაცვა; ***4)*** რამდენიმე მომხმარებლის მიერ ერთსა და იმავე ფაილზე ერთობლივი მუშაობის იმგვარად ორგანიზება, რომ მათ ამისათვის სპეციალური ძალისხმევა არ დასჭირდეს; ***5)*** არასანქცირებული შეღწევებისაგან ფაილის დაცვა; ***6)*** მაღალი მწარმოებლურობის უზრუნველყოფა.

ფაილი არის მეორეულ მეხსიერებაში ჩაწერილი შეკავშირებული ინფორმაციის ***სახელდებული*** ნაკრებია. ფაილური სისტემა ოპერაციული სისტემის ყველაზე ხილული ნაწილია. იგი სისტემის ყველა მომხმარებელს უზრუნველყოფს როგორც მონაცემების, ისე პროგრა-

მების ონლაინურად შენახვისა და შეღწევის მექანიზმით. მომხმარებლის თვალთახედვით ფაილი გარე მეხსიერების ერთეულია, ე. ი. დისკში ჩაწერილი ინფორმაცია აუცილებლად კონკრეტულ ფაილში შედის. ფაილები მათი სახელებით იდენტიფიცირდება.

მისამართების ერთობლიობა წარმოქმნის ***სამისამართო სივრცეს,*** ფაილების სახელების ერთობლიობა კი – ***სახელების სივრცეს;*** სამისამართო სივრცის ელემენტი მისამართით, ხოლო სახელთა სივრცის ელემენტი – სახელით მოიძებნება; კერძოდ, მისამართებით მოიძებნება კომპიუტერის ოპერატიული მეხსიერების, ხოლო სახელებით – დისკური მეხსიერების ობიექტები.

2.2. ფაილების ლოგიკური და ფიზიკური ორგანიზაცია. ფაილების მისამართები

ფაილში მონაცემები გარკვეული სტრუქტურის სახით უნდა იყოს ორგანიზებული. ეს სტრუქტურა წარმოადგენს ბაზისს, რომელზე დაყრდნობითაც მუშავდება მონაცემების დამამუშავებელი პროგრამები. მონაცემების სტრუქტურის დაცვა ეკისრება ***გამოყენებას***, თუმცა დაცვის ფუნქციები გარკვეულწილად შეიძლება შეასრულოს ოპერაციულმა სისტემამაც შესარულოს.

არსებობს ფაილის არასტრუქურირებული და სტრუქტურირებული მოდელები. ***არასტრუქტურირებული ფაილის*** სტრუქტურირებასა და ინტერპრეტირასთან დაკავშირებული სამუშაოების წარმართვა ***გამოყენებას*** ეკისრება. ეს რამდენიმე ***გამოყენებას*** აძლევს საშუალებას ფაილი ერთმანეთს შორის გაიყოს და თითოეულმა მათგანმა თავისებურად მოახდინოს მასში არსებული მონაცემების სტრუქტურირება და ინტერპრეტირება. ასეთი მიდგომა ***UNIX*** სახის ოპერაციული სისტემისთვის გახდა პოპულარული და დღეისათვის ***ფართოდ გამოიყენება თანამედროვე ოპერაციულ სისტემებში.***

ფაილის მეორე – ***სტრუქტურირებული მოდელი*** ადრეულ ოპერაციულ (***OS/360, DEC RSX uVMS***) სისტემებში გამოიყენებოდა და დღეისთვის იშვიათად გვხდება. ამ შემთხვევაში ფაილის სტრუქტურის მხარდაჭერა ფაილურ სისტემას ეკისრება. მოცემულ შემთხვევაში ***გამოყენებას*** შეუძლია ფაილურ სისტემას ლოგიკური ჩანაწერების შეტანა-გამოტანის მოთხოვნებით მიმართოს. ფაილურ სისტემას ფა-

ილის სტრუქტურის შესახებ უნდა ჰქონდეს ნებისმიერი მოთხოვნილი ლოგიკური ჩანაწერის გამოსაყოფად საკმარისი ინფორმაცია. ფაილური სისტემა გამოყენებას აძლევს ჩანაწერთან შეღწევის საშუალებას, ხოლო ამ ჩანაწერში არსებულ მონაცემებს ამუშავებს ***გამოყენება.*** ასეთი მიდგომის განვითარების შედეგად იქნა დამუშავებული ***მონაცემთა ბაზების მართვის სისტემები*** (*Database management system*).

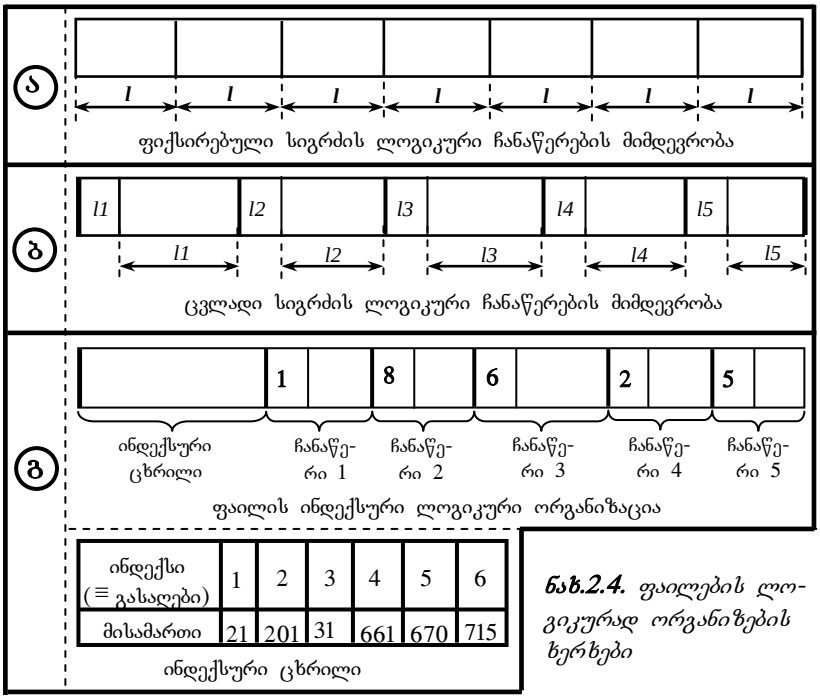
ერთმანეთისაგან განასხვავებენ ფაილების ლოგიკურ და ფიზიკურ ორგანიზაციას.***ლოგიკური ორგანიზაცია*** განსაზღვრავს ფაილში მონაცემების განთავსების პრინციპებს, ***ფიზიკური ორგანიზაცია*** კი გვიჩვენებს რეალურ ფაილებში როგორაა განხორციელებული ეს პრინციპები.

1. ფაილების ლოგიკური ორგანიზაცია

მონაცემები ფაილში მოწესრიგებული ლოგიკური ჩანაწერების სახით უნდა იყოს განთავსებული. ***ლოგიკური ჩანაწერი*** ეწოდება მონაცემთა უმცირესი ერთობლიობას, რომელზეც დამპროგრამებელი ოპერირებს გარე მოწყობილობასთან ინფორმაციის გაცვლისას. გარე მოწყობილობა მონაცემებს ოპერატიულ მეხსიერებასთან ცვლის ბლოკების სახით, რომელთა სიდიდე შეიძლება არ ემთხვეოდეს ლოგიკური ჩანაწერის სიდიდეს. მიუხედავად ამისა ოპერაციული სისტემა დამპროგრამებელს ლოგიკურ ჩანაწერებთან შეღწევის საშუალებას აძლევს.

განასხვავებენ ლოგიკურ ჩანაწერთან მივდევრობით და პირდაპირ (თავისუფალი) შეღწევას. ***მიმდევრობითი შეღწევის*** დროს მოცემულ ლოგიკურ ჩანაწერთან შეღწევისათვის აუცილებელია ამ ჩანაწერის წინ არსებული ყველა ჩანაწერის მიმდევრობით წაკითხვა. ***პირდაპირი (თავისუფალი) შეღწევის*** დროს კი ეს აუცილებელი არ არის – საჭირო ჩანაწერთან უშუალოდ შეიძლება შეღწევა.

ფაილზე ლოგიკური ჩანაწერების განთავსებას ფაილია ლოგიკურად ორგანიზება ეწოდება. განასხვავებენ ფაილის ლოგიკურად ორგანიზების შემდეგ სამ ხერხს: ***1)*** ფაილზე ფიქსირებული სიგრძის მიმდევრობის განთავსების ხერხი (ნახ. ***2.4,ა***); ***2)*** ფაილზე ცვლადი სიგრძის მიმდევრობის განთავსების ხერხი (ნახ. ***2.4,ბ***); ***3)*** ფაილის ლოგიკური ჩანაწერების განთავსება ინდექსური ცხრილის გამოყენებით (ნახ. ***2.4,გ***);



ნახ.2.4. *ფაილების ლოგიკურად ორგანიზების ხერხები*

პირველი ხერხის დროს ლოგიკურ ჩანაწერთა სიგრძეები ფაილის ფარგლებში ***უცვლელია.*** ამ შემთხვევაში ლოგიგიკურ ***n***–ურ ჩანაწერთან შეღწევა შეიძლება ყველა წინმდებარე ჩანაწერის მიმდევრორობით წაკითხვის გზით (მიმდევრობით შეღწევა), ან ჩანაწერის რიგითი ნომრის დახმარებით მისი გამოთვლილი მისამართით.

მეორე ხერხისდროს ჩანაწერთა სიგრძეები ფაილის ფარგლებში იცვლება. ამ შემთხვევაში გამოიყენება ლოგიკურ ჩანაწერთან მიმდევრობითი შეღწევის ხერხი. ჩანაწერებთან თავისუფალი შეღწევა შეუძლებელია. ფაილებს, რომელთა ჩანაწერებთან შეღწევა ხორციელდება მიმდევრობით, მათი პოზიციების ნომრების მიხედვით, ***არაინდექსირებული ანუ მიმდევრობითი ფაილები*** ეწოდება.

მესამე ხერხის დროს ვიღებთ ე. წ. ***ინდექსირებული ფაილებს,*** რომელთა ლოგიკურ ჩანაწერებთან თავისუფალი ხეღწერაა შესაძლებელი. ასეთ ფაილებს აქვს ერთი ან რამდენიმე ***საგასაღებო ველი,*** რომელთა მნიშვნელობების მითითების გზით შესაძლებელია ჩანაწე-

რების დამისამართება. მონაცემების სწრაფად მოსაძებნად ინდექსირებულ ფაილში გაითვალისწინება ***ინდექსური ცხრილი***, რომელშიც საგასაღებო ველების მნიშვნელობებს უთანადდება გარე მეხსიერების მისამართი. ეს მისამართი შეიძლება მიუთითებდეს ან საძებნ ჩანაწერს (***პირდაპირი შელწევა***), ან გარე მეხსიერების უბანს, რომელზედაც განთავსებულია მონაცემების ჯგუფი და ამ ჯგუფში შედის საძებნი ჩანაწერი. ამ შემთხვევაში ფაილი ***ინდექსურ-მიმდევრობითადაა ორგანიზებული***. საინდექსო ცხრილებს ქმნის ფაილური სისტემა.

2. ფაილების ფიზიკური ორგანიზაცია

ფაილის ფიზიკური ორგანიზაცია აღწერს გარე მეხსიერების მოწყობილობაზე ფაილის განთავსების წესებს (დისკზე ფაილის განთავსების წესებს (დისკზე ფაილის განთავსების ხერხს). ფაილების ფიზიკური ორგანიზაციის ხერხის ეფექტურობის კრიტერიუმებია: ***1)*** მონაცემებთან შელწევის სისწრაფე; ***2)*** ფაილის სამისამართო სივრცის მოცულობა; ***3)*** დისკური სივრცის ფრაგმენტირების ხარისხი; ***4)*** ფაილის შესაძლო მაქსიმალური სიდიდე.

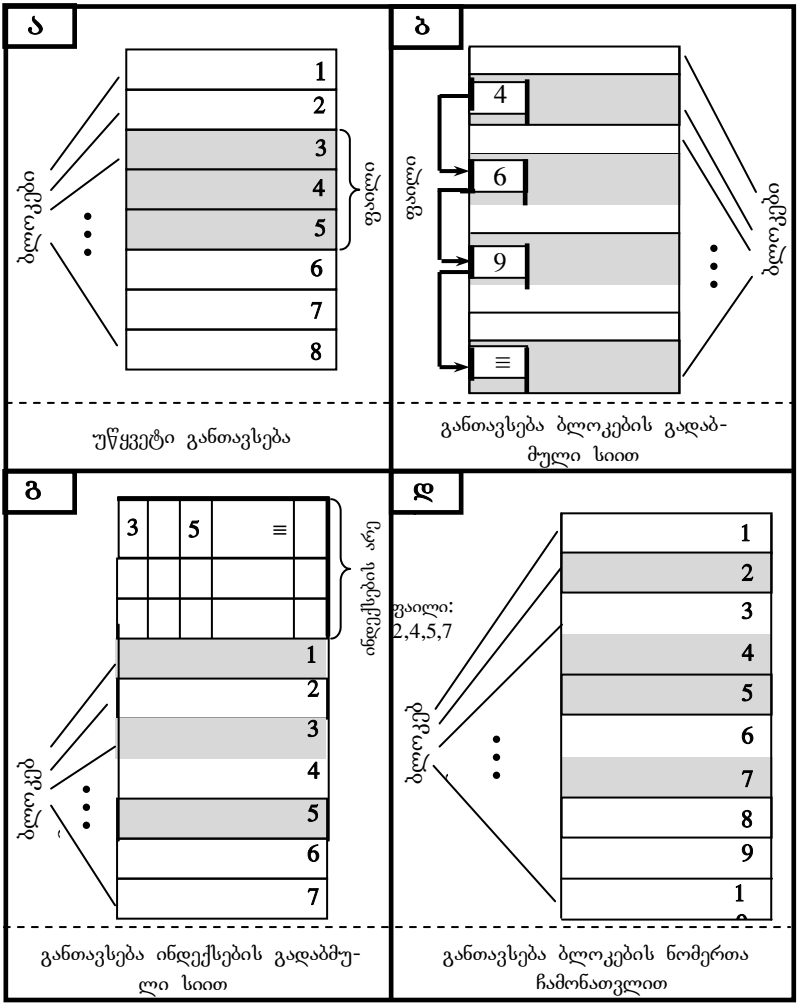
ფაილი შედგება ფიზიკური ჩანაწერებისაგან, რომლებსაც ***ბლოკები*** ეწოდება (ნახ. ***2.5***). ბლოკი მონაცემების უმცირესი ერთეულია, რომლებითაც გარე მეხსიერება ინფორმაციას ცვლის ოპერატიულ მეხსიერებასთან. არსებობს დისკის ბლოკებში ფაილების განთავსების ოთხი ვარიანტი: უწყვეტი განთავსება; განთავსება ბლოკების გადაბმული სიით; განთავსება ინდექსების გადაბმული სიით და განთავსება ფაილის მიერ დასაკავებელი ბლოკების ნომერთა უბრალო ჩამოთვლით. განვიხილოთ ისინი.

ა) უწყვეტი განთავსება (ნახ. ***2.5,ა***)დისკზე ფაილების განთავსების უმარტივესი ხერხია. ამ დროს ფაილს ეთმობა დისკის მიმდევრობით განთავსებული ბლოკები, რომლებიც დისკური მეხსიერების ერთიან უწყვეტ უბანს წარმოქმნის. ხერხს აქვს ორი ღირსება და ორი ნაკლი. ***ღირსებებია*** მისამართის ადვილად მოცემის შესაძლებლობა (ამისათვის ფაილის პირველი ბლოკის ნომრისა და ბლოკების რაოდენობის მითითებაა საკმარისი) და სტუქტურის სიმარტივე. ***ნაკლოვანებებია:*** ***ა)*** წინაწარ უცნობია ფაილის სიგრძე, ამიტომ უცნობია დისკის მისთვის დასარეზერვებელი სიდიდის მნიშვნელობაც; ***ბ)*** დისკის სივრცის არაეფექტური გამოყენება, რაც ფრაგ-

მენტირებას განაპირობებს: მცირე (*1* ბაიტზე ნაკლები) ზომის ცალკეული უბნები გამოუყენებელი რჩება.

უწყვეტი განთავსება გამოიყენება ოპერაციულ ***IBM/CMS, RSX-11*** და ზოგიერთ სხვა სისტემაში.

უწყვეტი განთავსება გამოიყენება ოპერაციულ ***IBM/CMS, RSX-11*** და ზოგიერთ სხვა სისტემაში.



ნახ. 2.5. *ფაილების ფიზიკურად ორგანიზების*

ბ)განთავსება დისკური მეხსიერების გადაბმული სიის მეშვეობით (ნახ.***2.5,ბ***). ასეთი ხერხის დროს თითოეული ბლოკი შეიცავს მომდევნო ბლოკზე მითითებას. ამ შემთხვევაშიც ფაილის მისამართი შეიძლება მხოლოდ ერთი რიცხვით – პირველი ბლოკის ნომრით – იქნას მოცემული. წინა წესისაგან განსხვავებით თითოეული ბლოკი შეიძლება მიმდევრობით მიუერთდეს ნებისმიერ მომდევნო ბლოკს, რის გამოც ფრაგმენტირება გამოირიცხება. ფაილის სიგრძე თავისი არსებობის პერიოდში შეიძლება შეიცვალოს დამატებითი ბლოკების მიერთების გზით. ***ნაკლია*** ფაილის ნებისმიერ ადგილზე შეღწევის პროცესის რეალიზაცია სირთულე; მაგალითად, მეხუთე ბლოკთან შესაღწევად აუცილებელია პირველი ოთხი ბლოკის წაკითხვა. გარდა ამისა, ფაილის ის ნაწილი, რომელივ შედის ბლოკში, არ არის **2**-ის ხარისხის ტოლი (ერთი სიტყვა იხარჯება ბლოკის ნომრის მისათითებლად), ხოლო მრავალი პროგრამა მონაცემებს კითხულობს ***2***-ის ხარისხის ტოლ ბლოკებად.

გ) განთავსება ინდექსების გადაბმული სიის მეშვეობით (ნახ. ***2.5,გ***). ფაილურ სისტემა ***FAT***–ში პოპულარულია ***ფაილების ფიზიკურად განთავსების ხერხი ინდექსების გადაბმული სიის გამოყენებით*** (ნახ. ***2.5,გ***). იგი წინა ხერხის გარკვეული მოდიფიცირების შედეგადაა მიღებული. ფაილს მეხსიერება ასევე ბლოკების შეკრული სიის სახით გამოეყოფა. პირველი ბლოკის ნომერი დამახსოვრებულია კატალოგის ჩანაწერში, რომელშიც ამ ფაილის მახასიათებლებია შენახული. დანარჩენი სამისამართო სივრცე გამოყოფილია ფაილის ბლოკებისაგან. დისკის თითოეულ ბლოკთან დაკავშირებულია გარკვეული ელემენტი – ***ინდექსი***. ინდექსები დისკის განცალკევებულ არეშია განთავსებული.

დ)ფაილისგანთავსება მის მიერ დასაკავებელიბლოკებისნომრითაუბრალოჩამოთვლით.ოპერაციული***UNIX*** სისტემა იყენებს მოცემული ხერხის ერთ-ერთ ვარიანტს, რომელიც საშუალებას იძლევა დამოუკიდებლად ფაილის ზომისა მისამართს ჰქონდეს ფიქსირებული სიგრძე. ფაილის მისამართის შესანახად ***13*** ველია გამოყოფილი. ფაილის ზომა თუ ***10*** ბლოკზე ნაკლებია, ან მისი ტოლია, მაშინ ამ ბლოკების ნომრები მისამართის პირველ ***10*** ველშია ჩამოთვლილი. ფაილის ზომა თუ აჭარბებს **10**-ს, მაშინ მე-***11*** ველი შეიცავს იმ ბლოკის მისამართს, რომელშიც შეიძლება ფაილის მომდევ-

ნო ბლოკების კიდევ ***128*** ნომერი იქნეს შენახული. ფაილის ზომა თუ (***10+128***)-ზე მეტია, მაშინ გამოიყენება ***128***-ე ველი, რომელშიც თავსდება ისეთი ***128*** ბლოკის ნომრების შემნახველი ბლოკის ნომერი, რომელთაგანაც თითოეულ ბლოკში ***128-128*** ბლოკის ნომრებია შენახული. საბოლოოდ ვიღებთ, რომ ასეთი სამმაგი ირიბი დამისამართების მეშვეობით მოცემული ხერხი საშუალებას გვაძლევს შევინახოთ ფაილი, რომლის მაქსიმალური ზომაა:

10+128 +128(128+128(128(128.

3. ფაილთან შეღწევის წესები.

ფაილთან შეღწევის წესის განსაზღვრა ნიშნავს თითოეული მომხმარებლისათვის განისაზღვროს ოპერაციების ნაკრები, რომლებიც მან შეიძლება შეასრულოს ამ ფაილზე. სხვადასხვა ფაილურე სისტემაშიშეიძლება განსაზღვრული იყოს შეღწევის დიფერენცირებულ ოპერაციათა საკუთარი სია. მასში შეიძლება შევიდეს ფაილისა და კატალოგების შექმნის, განადგურების, გახსნის, დახურვის, ფაილში ჩაწერის, ფაილის დამატების, ფაილში ძიების, ფაილის ატრიბუტების მიღების, ახალი ატრიბუტების დამატების, სახელის გადარქმევის, შესრულების, კატალოგის წაკითხვის და სხვა ოპერაციები.

უზოგადეს შემთხვევაში შეღწევის უფლებები შეიძლება აღიწეროს ***შეღწევის უფლებათა მატრიცით***. მასში თითოეული ფაილისათვის გამოყოფილია საკუთარი სვეტი, ხოლო თითოეული მომხმარებლისთვის – საკუთარი მწკრივი. სტრიქონებისა და მწკრივების გადაკვეთაზე მითითებულია რომელი ოპერაციაა ნებადართული (ნახ. ***2.6***). ზოგიერთ სისტემაში მომხმარებლები შეიძლება

ფაილთა სახელები

მომხმარებელთა სახელები	modem.tx	win.exe	class.d	unix.
G	წაკითხვა	შესრულება	-	შესრულება
S	წაკითხვა	შესრულება	-	შესრულება წაკითხვა
Dod	წაკითხვა	-	-	შესრულება წაკითხვა
V	წაკითხვა, ჩაწერა	-	შექმნა	-

ნახ.2.6. *შეღწევის უფლებათა მატრიცა*

ცალკეულ კატეგორიებად იყოს დაყოფილი. ერთი და იმავე კატეგორიის მომხმარებლებისათვის შეღწევის საერთო წესებია განსაზღვრული. მაგალითად, ***UNIX*** სისტემაში მომხმარებელთა მთელი სიმრავლე იყოფა სამ კატეგორიად: ფაილის მფლობელებად, ფაილის

მფლობელთა ჯგუფის წევრებად და ყველა სხვა მომხმარებლებად. განასხვავებენ შეღწევისადმი შემდეგ ორ მიდგომას:

1.ამორჩევით მიდგომას, რომლის დროსაც მფლობელს თავად შეუძლია განუსაზღვროს დასაშვები ოპერაციები თითოეულ ფაილსა და თითოეულ მომხმარებელს;

2.სამანდატო მიდგომას, რომლის დროსაც იმაზე დამოკიდებულებით, თუ რომელ ჯგუფს ეკუთვნის მომხმარებელი, სისტემა განუსაზღვრავს მფლობელს, თუ რა უფლება აქვს მას გასანაწილებელ რესურსზე (მოცემულ შემთხვევაში, ფაილზე).

4. დისკების კეშირების პრინციპები

ზოგიერთ ფაილურ სისტემაში იმ გარე მოწყობილობებისკენ გაგზავნილი მოთხოვნებს, რომლებშიც დამისამართება ბლოკებად ხდება (ასეთებია დისკები და ლენტები), ხელში იგდებს საშუალედო პროგრამული ფენა – ბუფერიზაციის ქვესისტემა. ***ბუფერიზაციის ქვესისტემას*** წარმოქმნის ოპერატიულ მეხსიერებაში განთავსებული ბუფერული პული და ამ პულის მმართველი პროგრამების კომპლექსი. ***პული*** ინფორმატიკაში ეწოდება გამოსაყენებლად მზა ობიექტების ნაკრებს.

პულის თითოეულ ბუფერის ზომა ერთი ბლოკის ტოლია. გარკვეული პროცესიდან რომელიმე ბლოკის წაკითხვის მოთხოვნის მოსვლისას ***ბუფერიზაციის ქვესისტემა*** გადაათვალიერებს საკუთარ ***ბუფერულ პულს.*** მასში შეიძლება აღმოჩნდეს ან არ აღმოჩნდეს აღნიშნული ბლოკი. ***პირველ შემთხვევაში*** აღმოჩენილ ბლოკს ბუფერიზაციის ქვესისტემა პირდაპირ გადააკოპირებს მოთხოვნის გამგზავნი პროცესის ბუფერში და ამით შეტანა/გამოტანის ოპერაცია ჩაითვლება დამთავრებულად, თუმცა უშუალოდ მოწყობილობიდან ინფორმაცია ფიზურად არ გაცვლილა. ამით ვიგებთ დროს, რომელიც უნდა დახარჯულიყო ფაილთან შეღწევისათვის. ***მეორე შემთხვევაში*** ბუფერული სისტემა მოწყობილობიდან წაიკითხავს მოთხოვნილ ბლოკს და მომთხოვნ პროცესისათვის მის გადაგზავნასთან ერთდროულად მოახდენს მის კოპირებას ***ბუფერიზაციის ქვესისტემის ერთერთ ბუფერში.*** თავისუფალი ბუფერის არარსებობისას დისკზე გაიტანება ყველაზე ნაკლებად გამოყენებადი ინფორმაცია და მის ადგილზე ჩაიწერება მოთხოვნილი ბლოკი. მაშასადამე, ბუფერიზაციის ქვესისტემა კეშ-მეხსიერების პრინციპის მიხედვით მუშაობს. აღნიშ-

ნული პრინციპი [2]–ში გვაქვს გადმოცემული. აქ კი დაინტერესებულ პირებს კეშის შესახებ დამატებით ინფორმაციას მივაწვდით, რომელიც მათ თვალსაწიერის გაფართოებაში დაეხმარება.

კეშს *(ინგლ.* ***cach,*** *ფრანგ.* ***cacher*** *– „დამალვა“) გამოთვლით ტექნიკაში უწოდებენ მონაცემების შესანახად განკუთვნილ სწრაფად შეღწევად ბუფერს.* ***ბუფერად*** *კი, თავის მხრივ, სხვა არაფერია, თუ არამეხსიერების გარკვეული მიდამო, რომელშიც შენახული მონაცემები შეიძლება გაიცვალოს როგორც გარე მოწყობილობებთან, ასევე პროცესებთანაც. ბუფერი შეიძლება რეალიზებული იყოს როგორც აპარატურულად, ასევე პროგრამულადაც, მაგრამ უმეტეს შემთხვევაში ისინი პროგრამულადაა რეალიზებული. ბუფერები მაშინ გამოიყენება, როდესაც მონაცემების მიღების სიჩქარე მნიშვნელოვნად განსხვავდება მათი დამუშავების სიჩქარისაგან, ან როდესაც ამ სიჩქარეებს ცვლადი მნიშვნელობები აქვს (მაგალითად, ბეჭდვის ბუფერიზაცია) და ა.შ.*

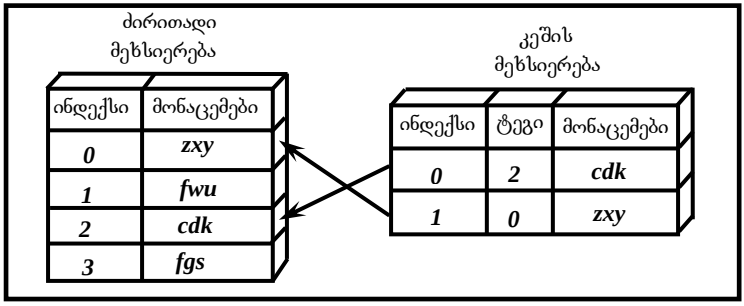
ტერმინები ***კეში*** *და* ***ბუფერი*** *არაა ურთიერთგამომრიცხავი ტერმინები, რის გამოც ხშირად მათ ფუნქციებს ერთმანეთში ურევენ. კლასიკური გაგებით* ***ბუფერი*** *არის დროებითი საცავი, სადაც მონაცემთა დიდი ბლოკები ერთმანეთს შეერევა, ან ცალკეულ ნაწილებად დაიყოფა. რაც შეეხება* ***კეშს****, იგი ისეთი* ***სპეციფიკური ბუფერია****, რომელშიც მონაცემები უფრო ხშირად წაიკითხება, ვიდრე ჩაიწერება.* ***კეშის შექმნის მიზნია:*** *შევამციროთ დამხსომებელ მოწყობილობებთან მიმართვების რაოდენობა და ეს მიმართვები უფრო ეფექტური გავხადოთ. კეშში შენახულ მონაცემებთან შეღწევა უფრო სწრაფად ხდება, ვიდრე ნელმომქმედი საცავიდან მონაცემების ამოკრება; ოღონდ კეშის მოცულობა საწყისი მონაცემების საცავის მოცულობაზე გაცილებით მცირეა.*

ტერმინი „კეში“კომპიუტერულ მეცნიერებაში ჟურნალ „IBM Systems Journal“-ის ძალისხმევით იქნა შემოტანილი. მისმა რედაქტორმა ***ლეილ ჯონსონმა1967*** *წელს მკითხველებს შესთავაზა ტერმინ „მაღალჩქაროსნული ბუფერის“ ნაცვლად მოეფიქრებინათ კომპაქტური შესატყვისი; შემოთავაზებების არარსებობის გამო* ***1968*** *წელს თვითონ მოიფიქრა ტერმინი „cache“(კეში). მან მოწონება დაიმსახურა და კომპიუტერულ ლიტერატურაში მყარად დამკვიდრა. აღნიშნულიდან გამომდინარე ტერმინები „მაღალჩქაროსნული ბუფერი“ და „კეში“ სინონიმებია.*

კეში *ეს არის* ***ძირითად მეხსიერებად*** *წოდებულ დაბალი შეღწევადობის მეხსიერებაში არსებულ მონაცემებთან ურთიერთობის სწრაფად დამყარებისათვის განკუთვნილი* ***ზესწრაფად შეღწევადი მეხსიერება****. კეშირება გამო-*

იყენება ცენტრალური პროცესორული მოწყობილობებისათვის, ხისტი დისკებისათვის, ბრაუზერებისათვის და ა. შ.

*კეში შედგება ჩანაწერების ნაკრებისაგან (ნახ. **2.7**). თითოეული ჩანაწერი ასოცირებულია მონაცემთა ელემენტთან (მონაცემების პატარა ნაწილთან), რომელიც წარმოადგენს ძირითად მეხსიერებაში არსებულ მომაცემთა ელემენტის ასლს. თითოეულ ჩანაწერს აქვს **ტეგად** წოდებული იდენტიფიკატორი. იგი განსაზღვრავს კეშში არსებული ჩანაწერი ძირითად მეხსიერებაში არსებულ მონაცემთა რომელ ელემენტს შეესაბამება (ტეგი – **tag** – ინგლისური ენიდან ნასესხები სიტყვაა, რომელიც, ფართო გაგებით, **ჭდე** ანუ **მარკირების ნიშანია**)*



ნახ. 2.7. *ძირითად მეხსიერებაში კეშის მეხსიერების ასახვა*

*კეშის კლიენტი (ცენტრალური პროცესორი, ბრაუზერი, ოპერაციული სისტემა) მონაცემებთან მიმართვისას, უპირველეს ყოვლისა, ათვალიერებს კეშს. იქ ისეთი ჩანაწერის აღმოჩენისას, რომლის იდენტიფიკატორი ემთხვევა მონაცემთა მოთხოვნილი ელემენტის იდენტიფიკატორს, კლიენტი იყენებს კეშში არსებულ მონაცემებს. ამას **კეშის მიზანში მოხვედრა** ეწოდება. ასეთი ჩანაწერის არარსებობისას კლიენტი პროცესი იძულებული ხდება მეტი დრო დახარჯოს და მონაცემები უშუალოდ მოწყობილობიდან წაიკითხოს, რასაც **კეშის მიზნისათვის აცდენა** ეწოდება. წაკითხული მონაცემების ასლი დამატებით ჩაიწერება კეშში და ამით მომავალში კლიენტს მისი კეშიდან წაკითხვის საშუალება ექნება. კეშისადმი ისეთი მიმართვების რაოდენობას, რომელთა დროსაც ნაპოვნი იქნა სასურველი მონაცემები, **მიზანში მოხვედრის დონე** ანუ **კეშში მიზანში მოხვედრის კოეფიციენტი** ეწოდება.*

შეზღუდული მოცულობის კეშის დროს შეიძლება კეშიდან სივრცის გათავისუფლებისათვის მიღებული იქნას მასში არსებული ნაკლებად გამოყენებადი ჩანაწერების გაძევების გადაწყვეტილება; გასაძევებელი მონაცემები კეშში თუ ადრე იყო მოდიფიცირებული, ისინი გაძევებამდე გადაკოპირდება ძირითად მეხსისიერებაში და კეშიდან მხოლოდ ამის შემდეგ გაძევდება. სა-

წინაალმდეგო შემთხვევაში მათ მიერ დაკავებული ადგილები უბრალოდ თავისუფლად გამოცხადდება.

მონაცემების შესანახი მრავალი პერიფერიული მოწყობილობა, კერძოდ, ხისტი დისკები, იყენებს **1**-დან ***64*** მეგაბაიტამდე მოცულობის შინაგან კეშს; წასაკითხად გამოყენებული **CD/DVD/BD**-დისკებიც განმეორებითი მიმართვის ასაჩქარებლად ხშირად ახდენს წაკითხული ინფორმაციის კეშირებას.

ოპერაციული სისტემა ოპერატიული მეხსიერების ნაწილს იყენებს დისკური ოპერაციების კეშად იმ მოწყობილობებისათვის, რომლებსაც არ გააჩნია საკუთარი კეშმეხსიერება (ხისტი დისკებისათვის, **flash**-მეხსიერებისათვის, მოქნილი დისკებისათვის). ხშირად ხისტი დისკების კეშირებისათვის ოპერატიული მეხსიერების მთელი თავისუფალი ნაწილიც გამოიყენება.

გარე დამგროვებლებისათვის კეშირებას შემდეგი ფაქტორების გამო იყენებენ:

▲ ოპერატიულ მეხსიერებაში პროცესორის შეღწევის სიჩქარე ასჯერ და უფრო მეტად აღემატება გარე დამგროვებლებში მის შეღწევის სიჩქარეს.

▲ მონაცემების შენახვის დისკური მოწყობილობების (ხისტი, მოქნილი, ოპტიკური დისკების) მწარმოებლურობა რამდენიმე მიმდევრობით განთავსებული ბლოკების წაკითხვა-ჩაწერისათვისაა მაქსიმალური და მნიშვნელოვნად მცირდება დისკის სხვადასხვა ადგილისადმი ერთეულოვანი მიმართვის დროს; ამას განაპირობებს მექანიკური მოწყობილობებისათვის დამახასიათებელი ინერციულობა;

▲ გარე დამგროვებლების სხვადასხვა ბლოკისადმი მიმართვის სიხშირის უკიდურესი არათანაბარია;

▲ ბლოკების ნაწილს ჩასაწერად და წასაკითხად ერთდროულად რამდენიმე პროცესი იყენებს;

▲ ბლოკების ნაწილი ძალიან ხშირად წაიკითხება;

▲ ბლოკების ნაწილში ძალიან ხშირად ხდება ჩაწერა;

წაკითხვისას კეში ბლოკის მხოლოდ ერთხელ წაკითხვის საშუალებას იძლევა, რომლის შემდეგ ყველა პროცესის გამოსაყენებლად ბლოკის ერთ ასლს ინახავს და მათ შიგთავსს „მყისიერად“ აწვდის (დისკიდან მიწოდებასთან შედარებით). არსებობს „წინასწარ მოთხოვნის“ მექანიზმიც – ფონურ რეჟიმში ოპერაციული სისტემა კეშში წაიკითხავს საჭირო ჩანაწერის მომდევნო რამდენიმე ბლოკს.

ჩაწერის დროს კეში საშუალებას იძლევა მოკლე ჩანაწერები დაჯგუფდეს უფრო მსხვილ ჩანაწერებად, რომლებსაც უფრო ეფექტურად დაამუშავებს დამგროვებლი, ან რაც საშუალედო მოდიფიკაციების ჩაწერის თავიდან

აცილების საშუალებას იძლევა. ამ დროს პროცესორი ხედავს ოპერატიულ მეხსიერებაში არსებულ ბლოკის ყველა საშუალედო მდგომარეობას.

გარე შემნახავი მოწყობილობების კეშირებისას შეტანა-გამოტანის ოპტიმალურად გამოყენების გამო მნიშვნელოვნად იზრდება სისტემის მწარმოებლურობა. ტექნოლოგიის უპირატესობაა ფაილებთან მომუშავე გამოყენების უცვლელობის დროს დისკების მეხსიერების გამოყენების ოპტიმიზაციის გამჭვირვალეობა.

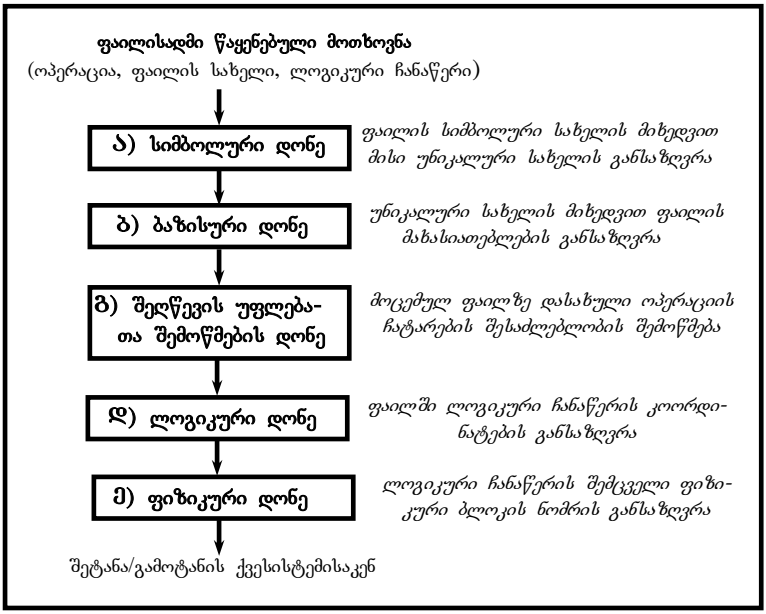
*ჩანაწერის კეშირების ნაკლია ის, რომ პროგრამისაგან ჩაწერაზე გაკეთებულ მოთხოვნასა და დისკზე ბლოკის ფაქტობრივ ჩაწერას შორის არსებობს დროის გარკვეული მონაკვეთი, ავრეთვე იცვლება ჩანაწერების შესრულების წესი, რამაც შეიძლება გამოიწვიოს ინფორმაციის დაკარგვა, ან სისტემის „დაკიდება“. მოცემული პრობლემა იძულებითი პერიოდული სინქრონიზებითა და ჟურნალირებადი ფაილური სისტემის გამოყენებით შეიძლება შერბილდეს (**ჟურნალირებადი** ეწოდება ფაილურ სისტემას, რომელიც სპეციალურად ფორმირებულ ჟურნალში შეინახავს ცვლილებათა სიას, რაც ამოვარდნების დროს ფაილური სისტემის მთლიანობის შენარჩუნებას უწყობს ხელს).*

5. ფაილური სისტემის ფუნქციონირების საფუძვლები

ნებისმიერი ფაილური სისტემის ფუნქციონირება შეიძლება წარმოვიდგინოთ მრავალდონიანი მოდელის სახით, რომელშიც თითოეული დონე წარმოადგენს ზევით მდებარე დონის ინტერფეისს (ფუნქციების ნაკრებს), ხოლო თავად ეს დონე საკუთარი ფუნქციონირებისათვის ინტერფეისად ქვემოთ მდებარე ინტერფეისს იყენებს (მას მიმართავს მოთხოვნებით). აღნიშნული მოდელი შედგება სიმბოლური, ბაზისური, შეღწევის უფლებათა შემოწმების, ლოგიკური და ფიზიკური დონეებისაგან (ნახ. ***2.8***). მოკლედ განვიხილოთ თითოეული მათგანი.

ა) სიმბოლური დონისამოცანაა ფაილის სიმბოლური სახელის მიხედვით განსაზღვროს მისი უნიკალური ნომერი. ერთადერთი სიმბოლური სახელის მქონე ფაილებისაგან შემდგარ სისტემებში (როგორიცაა, მაგალითად, ***MS-DOS***) ეს დონე არ არსებობს, რადგან ფაილისათვის მომხმარებლის მიერ მინიჭებული სიმბოლური სახელი თავისთავად უნიკალურია და იგი შეიძლება გამოიყენოს ოპერაციულმა სისტემამ. ფაილურ სისტემებში, რომლებშიც ერთსა და იმავე ფაილს შეიძლება რამდენიმე სიმბოლური სახელი ჰქონდეს, ფაილის უნიკალური სახელის განსასაზღვრავად მოცემულ დონეზე თვალიერ-

დება კატალოგების მწკრივი. ფაილურ ***UNIX*** სისტემაში უნიკალურ სახელად მიღებულია ***ფაილის ინდექსური დესკრიპტორის ნომერი*** ანუ ***i-node***.



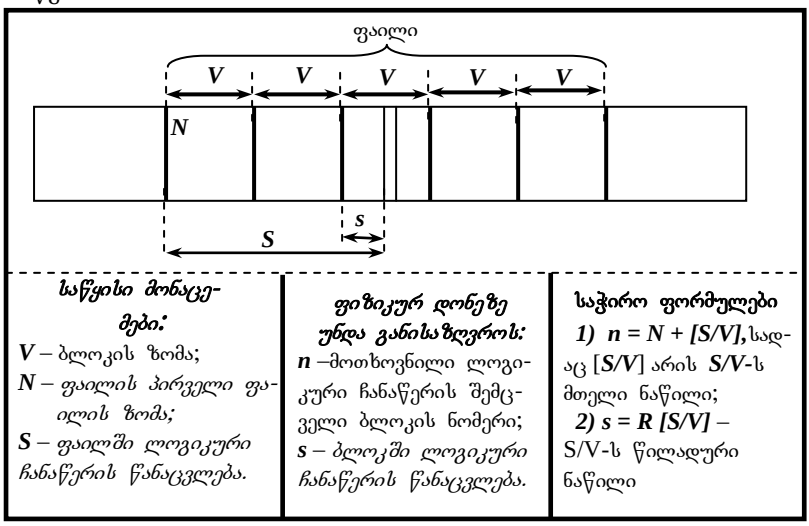
ნახ. 2.8 *ფაილური სისტემის ზოგადი მოდელი*

<u>ბ) ბაზისურ დონეზე</u>უნიკალური სახელითგანისაზღვრება ფაილის მახასიათებლები: შეღწევის უფლებები, მისამართი, ზომა და ა.შ. ისინი შეიძლება შედიოდეს კატალოგში ან ინახებოდეს განცალკევებულ ცხრილებში. ფაილთან შეღწევის საშუალო დროის შესამცირებლადფაილის გახსნისას მახასიათებლები დისკიდან ოპერატიულ მეხსიერებაში გადაიტანება. ზოგიერთ ფაილურ სისტემაში (მაგალითად,***HPFS***-ში) ფაილის გახსნისას დისკიდან ოპერატიულ მეხსიერებაში აღნიშნულ მახასიათებლებთან ერთად ფაილის მონაცემების შემცველი რამდენიმე საწყისი ბლოკიც გადაიტანება.

<u>გ) შეღწევის უფლებათა შემოწმების დონეზე</u>მოწმდება მოწმხმარებელს ან პროცესს ჰქონდა თუ არა მოცემული მოთხოვნის გაცემის უფლება. შეღწევის უფლებათა მატრიცით (იხ. ცხრ. ***2.6***) ამის დადასტურებისას პროცესი გაგრძელდება, საწინააღმდეგო შემ-

თხვევაში კი გაიცემა შეღწევის უფლებების დარღვევის შეტყობინება.

<u>დ) ლოგიკურ დონეზე</u> განისაზღვრებაფაილიდან მოთხოვნილი ლოგიკური ჩანაწერის კოორდინატები, კერძოდ, დგინდება მოთხოვნილი ჩანაწერი რა მანძილითაა (რამდენი ბაიტითაა) დაშორებული ფაილის დასაწყისიდან. ამ დროს ყურადღება არ ექცევა ფაილის ფიზიკურ განთავსებას: იგი ბაიტების უწყვეტი მიმდევრობის სახით წარმოიდგინება. მოცემული დონის მუშაობის ალგორითმი ფაილის ლოგიკურ ორგანიზაციაზეა დამოკიდებული. მაგალითად, ფაილი თუ ***1*** ბაიტი სიგრძის ლოგიკური ჩანაწერების მიმდევრობას წარმოადგენს, მაშინ ***n***-ური ლოგიკური ჩანაწერი ფაილის დასაწყისიდან (***n - 1***) ბაიტის ტოლი მანძილით იქნება დაშორებული ანუ წანაცვლებული. ***ინდექსურ-მიმდევრობითი ორგანიზაციის*** მქონე ფაილში ლოგიკური ჩანაწერის კოორდინატების გასაგებად წაიკითხება ინდექსების (გასაღებების) ცხრილი, რომელშიც უშუალოდაა მითითებული ლოგიკური ჩანაწერის მისამართი.



ნახ. 2.9. *ფაილური სისტემის ფიზიკური დონის ფუნქციები*

<u>ე) ფიზიკურ დონეზე</u> ფაილური სისტემა განსაზღვრავს მოთხოვნილი ლოგიკური ჩანაწერის შემცველი ფიზიკური ბლოკის ნომერს და ამ ბლოკში საძებნი ჩანაწერის წანაცვლებას (ბლოკის და-

საწყისიდან ამ ჩანაწერის დაშორების მანძილს). მოცემული ამოცანის გადასაწყვეტად გამოიყენება ლოგიკურ დონეზე გადაწყვეტილი ამოცანის შედეგები: ფაილში ლოგიკური ჩანაწერის წანაცვლება, გარე მოწყობილობაზე ფაილის მისამართი, აგრეთვე ცნობები ფაილის ფიზიკური ორგანიზაციის შესახებ (ბლოკის ზომის ჩათვლით). უწყვეტი მიმდევრობითი ბლოკების სახით რეალიზებული უმარტივესი ფაილისათვის ფიზიკურ დონეზე შესრულებული სამუშაოები ***2.9*** ნახაზზეა ილუსტრირებული. უნდა აღვნიშნოთ, რომ ფიზიკური დონის ამოცანა წყდება ფაილის ლოგიკური ორგანიზაციისაგან დამოუკიდებლად.

ფიზიკური ბლოკის ნომრის განსაზღვრის შემდეგ ფაილური სისტემა გარე მოწყობილობასთან ინფორმაციის გასაცვლელად მიმართავს შეტანა/გამოტანის სისტემას. ამ მიმართვის საპასუხოდ ფაილური სისტემის ბუფერში გადაიცემა საჭირო ბლოკი, რომელშიც ფიზიკურ დონეზე მიღებული წანაცვლების საფუძველზე ამოირჩევა მოთხოვნილი ლოგიკური ჩანაწერი.

6. მეხსიერებაზე ფაილე-ბის ასახვა

მეხსიერებაზე ფაილის ასახვა ფაილებთან მუშაობის ხერხია, რომლის დროსაც მთელ ფაილს ან მის რომელიმე უწყვეტ ნაწილსა და მეხსიერებას (ოპერატიული მეხსიერების მისამართების დიაპაზონს) შორის მყარდება შესაბამისობა (ნახ. ***2.10***). აღნიშნული შესაბამისობის დამყარების შემდეგ ამ მისამართებიდან მონაცემების წაკითხვას ფაქტობრივად მივყავართ ასახული ფაილიდან მონაცემების წაკითხვამდე, ხოლო ამ მისამართებზე მონაცემების ჩაწერას - ფაილში ამ მონაცემების ჩაწერამდე.

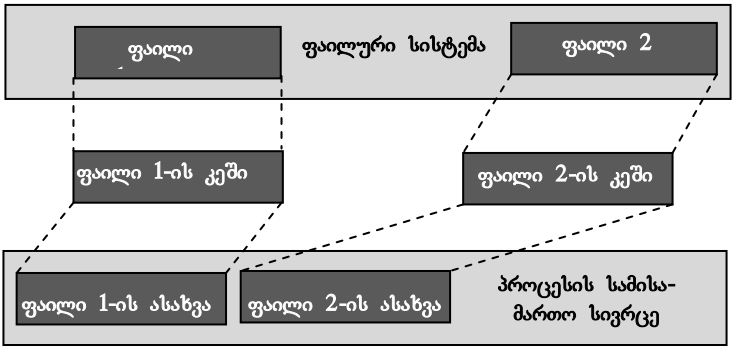
ამ ხერხის ალტერნატივას წარმოადგენს ფაილიდან პირდაპირი წაკითხვა ან ფაილში პირდაპირი ჩაწერა. მუშაობის ასეთი ხერხი ნაკლებად მოსახერხებელია შემდეგი მიზეზების გამო:

1. მუდამ უნდა გვახსოვდეს ფაილის მიმდინარე პოზიცია, საიდანაც უნდა მოხდეს წაკითხვა ან სადაც უნდა მოხდეს ჩაწერა;

2. მიმდინარე პოზიციის შეცვლა/ჩაწერისა და ჩაწერა/წაკითხვის თითოეული გამოძახება სისტემური გამოძახებაა, რომელიც იწვევს დროის დანაკარგს;

3. წაკითხვა/ჩაწერის მეშვეობით მუშაობისათვის მაინც გამოიყოფა გარკვეული ზომის ბუფერები, ე.ი. ზოგადად მუშაობა შედგება შემ-

დეგი სამი ეტაპისაგან: „ბუფერში წაკითხვა → ბუფერში მონაცემების მოდიფიცირება – ფაილში ჩაწერა“. ასახვის დროს კი მუშაობა კი მხოლოდ ერთი ეტაპისაგან შედგება: „მეხსიერების გარკვეულარ-ეში მონაცემების მოდიფიცირება“ .



ნახ. 2.10. *მეხსიერებაში ფაილის ასახვა*

მეხსიერებაზე ფაილების ასახვის ხერხის დამატებითი ღირსებაა ისიც, რომ წაკითხვა/ჩაწერასთან შედარებით იგი უფრო ნაკლებად ტვირთავს ოპერაციულ სისტემას; კერძოდ, ასახვის ხერხის გამოყენებისას ოპერაციული სისტემა მეხსიერებაში ფაილს არა ერთბაშად, არამედ მეხსიერების გვერდის (***4კბ***-ის) ზომის ცალკეული ბლოკების სახით ჩატვირთავს, ამიტომ მცირე (მაგ.***32 მბ***-ის) ზომის ფიზიკური მესიერებაში ადვილად და ზედნადები ხარჯების მნიშვნელოვნად გაუზრდელად აისახება დიდი (***100*** და უფრო მეტი მეგაბაიტის) ზომის ფაილი; ასევე ხდება მეხსიერებიდან დისკზე ჩაწერის დროსაც: მეხსიერებაში დიდი რაოდენობის მონაცემებს განახლებისას ისინი ერთდროულად (დისკზე წამკითხავი თავის ერთი გავლის პერიოდში) ჩაიწერება.

მეხსიერებაში ასახული ფაილი კიდევ იმითაა მოსახერხებელი, რომ შეიძლება ადვილად შეიცვალოს მისი ზომები და ხელახლად ასახვისათვის მივიღოთ მეხსიერების საჭირო ზომის მქონე უწყვეტი მონაკვეთი. ფრაგმენტაციის მოვლენის გამო ასეთ ტრიუკს დინამიკური მეხსიერების გამოყენებისას ყოველთვის ვერ შევასრულებთ, მაგრამ თუ მეხსიერებაში ასახულ ფაილთან გვიხდება მუშაობა, მა-

შინ მეხსიერების მენეჯერი პროცესორს იმგვარად ააწყობს, რომ ოპერატიული დამხსომებელი მოწყობილობის გვერდები, რომლებშიც შენახულია ფაილის მეზობელი ფრაგმენტები, წარმოქმნის მისამართების უწყვეტ დიაპაზონს.

მეხსიერებაზეფაილისასახვას ძირითადად მწარმოებლურობის ასამაღლებლად ვიყენებთ, მაგრამ ამ დროს უნდა გვახსოვდეს კომპრომისები, რომლებზეცშეიძლება მოგვიხდეს წასვლა. ***მონაცემების ჩვეულებრივი შეტანა-გამოტანის ხერხისგამოყენებისას*** სისტემური გამოძახებები და ზედმეტი კოპირებები წარმოშობს ზედნადებ ხარჯებს, ხოლო მეხსიერებაზე ფაილის ასახვის დროს – გვერდობრივმა შეცდომებმა შეიძლება შეანელოს პროცესი.

დავუშვათ, რომ საჭირო ფაილის კუთვნილი გვერდი უკვე დევს კეშში, მაგრამ არ არის ასოცირებული მოცემულ ასახვასთან. მას ამ დროს თუ შეცვლის სხვა პროცესი, მაშინ წარუმატებლად დასრულდება ასახვასთან მისი ასოცირების მცდელობა და იძულებული გავხდებით მონაცემები განმეორებით წავიკითხოთ დისკიდან ან შევინახოთ ისინი დისკზე. ამიტომ მიუხედავად იმისა, რომ ასახვის დროს მცირე რაოდენობის ოპერაციებია შესასრულებელი, რეალურად ფაილის რომელიმე ადგილზე მონაცემების ჩაწერას შეიძლება იმაზე მეტი დრო დასჭირდეს, ვიდრე ეს საჭიროა ფაილური შეტანა-გამოტანის ტრადიცული ოპერაციების გამოყენების დროს (ეს მაშინ, როდესაც საშუალოდ ასახვის გამოყენება მომგებიანია).

მეორე ნაკლი ისაა, რომ ასახვის ზომა გამოყენებულ არქიტექტურაზეა დამოკიდებული. ***32***-ბიტური არქიტექტურის დროს თეორიულად ***4*** გიგაბაიტზე უფრო დიდი ასახვის რეალიზება შეუძლებელია.

თანამედროვე ოპერაციული სისტემების ან გარსაცმების უმრავლესობა მხარს უჭერს მეხსიერებაში ასახულ ფაილებთან მუშაობის ამა თუ იმ ფორმას. მეხსიერებაში ასახულ ფაილებს ***MMF-ფაილები*** (ინგ. „***M****emory-****M****apped* ***F****iles*“; ქართ. ანუ „მეხსიერებაში ასახვადი ფაილები“), ხოლო მეხსიერებაში ფაილების ასახვის მექანიზმს ***MMF-მექანიზმი*** ეწოდება. ეს მექანიზმი პირველად გამოჩნდა ოპერტიულ სისტემა ***MULTICS***-ში და მასში შემოტანილი იქნა ორი ახალი სახის სისტემური გამოძახება: „***MAP***“ („აისახოს“) „***UNMAP***“ („აიკრძალოს ასახვა“). ***პირველი გამოძახება*** ოპერაციულ სისტემას პარამეტრების

სახით გადასცემს ფაილის სახელსა და ვირტუალურ მისამართს, სადაც უნდა აისახოს მოცემული ფაილი; ამის შემდეგ ოპერაციული სისტემა ვირტუალური სამისამართო სივრცის მითითებულ მისამართზე ასახავს აღნიშნულ ფაილს.

*ოპერაციული **MULTICS** სისტემა **1964-70** წლებშიიქმნა დამუშავებული(**MULT**iplexed **I**nformation and **C**omputing **S**ervice „მულტიპლექსური და გამოთვლითი სამსახური“). კომპიუტერულ ბაზარზე მისი გამოსვლა წარუმატებელიაღმოჩნდა,მაგრამ მასში ჩადებული მრავალი ნოვატორული და ღირებული იდეების გამო აღნიშნულმა სისტემამ ძალიან დიდი გავლენა მოახდინა კომპიუტერულ ინდუსტრიაში. კრიტიკოსებისაგან მიღებული მრავალი მწარე დაცინვის მიუხედავად **MULTICS**-ში მასში შემოთავაზებული მრავალი ინოვაცია ხანგრძლივად დამკვიდრდა ოპერაციულ სისტემათა სამყაროში ხანგრძლივად დამკვიდრდა. ერთ-ერთი ასეთი ნოვაცია იყო ე. წ. **MMF**-ფაილები და **MMF**-მექანიზმის რეალიზების საფუძველი.*

ახალი ოპერაციულ სისტემათა შემქმნელები ცდილობენ მომხმარებლებს რამდენიმე ფაილურ სისტემასთან მუშაობის საშუალება მისცენ. ახალი გაგების ფაილური სისტემა შეიცავს მრავალ მდგენელს, რომელთა შორის არის ტრადიციული გაგების ***ფაილური სისტემაც***.

ახალი ფაილური სისტემის სტრუქტურა მრავალდონიანია. ზედა დონეზე განთავსებულია ე.წ. ***ფაილური სისტემების გადამრთველი***. იგი წარმოქმნის ინტერფეისს პროგრამის მოთხოვნებსა და იმ ფაილურ სისტემას შორის, რომლისკენაცაა მიმართული ეს მოთხოვნები. ფაილური სისტემების გადამრთველი მოთხოვნებს გარდაქმნის მომდევნო დონისათვის აღსაქმელ ფორმატად.

ფაილური სისტემების დონის თითოეული კომპონენტი ამ სისტემის დრაივერის სახითაა ფორმირებული და განსაზღვრავს ფაილური სისტემის გარკვეულ სტრუქტურას. გადამრთველი ერთადერთი მოდულია, რომელსაც შეუძლია მიმართოს ფაილური სისტემის დრაივერს. ფაილური სისტემის დრაივერი შეიძლება დაიწეროს ***რეენტერაბელური კოდის*** სახით, რაც რამდენიმე პროგრამას საშუალებას აძლევს, ერთდროულად შეასრულოს ოპერაციები ფაილებთან. ფაილური სისტემის თითოეული დრაივერი საკუთარი ინიციალიზაციის პროცესის დროს რეგისტრირდება გადამრთველთან, გადასცემს რა მას შესასვლელის წერტილების ცხრილს, რომლებსაც გამოიყენებს

ფაილური სისტემასთან მომდევნო მიმართვების დროს. *[კომპიუტერულ პროგრამას ან მის ცალკეულ პროცედურას ეწოდება **რეენტერაბელური** (ინგ. reentrant –„განმეორებით შემავალი“), თუ იგი ისეა დამუშავებული, რომ რამდენიმე მომხმარებელს ან პროცესს შეუძლია მეხსიერებაში გამო-იყენოს პროგრამის ინსტრუქციის ერთი და იგივე ასლი. ამასთანავე, მეორე მომხმარებელს უნდა შეეძლოს მანამდე გამოიძახოს რეენტერაბელური კოდი, სანამ პირველი მომხმარებელი დაასრულებს მასზე მუშაობას და ამან, სულ მცირე, არ უნდა წარმოშვას შეცდომა, ხოლო კორექტული რეალიზაციის შემთხვევაში - არ უნდა გამოიწვიოს გამოთვლების დაკარგვა ე.ი. კოდის უკვე შესრულებული ფრაგმენტი ხელახლა შესასრულებელი არ გახდეს].*

საკუთარი ფუნქციების შესასრულებლად ფაილურ სისტემათა დრაივერები მიმართავს ახალი არქიტექტურის მქონე მომდევნო ფაილური სისტემის წარმომქმნელ შეტანა/გამოტანის ქვესისტემას.

7. შეტანა/გამოტანის ქვესისტემა

შეტანა/გამოტანის ქვესისტემა ფაილური სისტემს შემადგენელი ნაწილია, რომელიც პასუხისმგებელია ფაილური სისტემის ქვედა დონეთა ყველა მოდულის ჩატვირთვის, ინიციალიზაციასა და მართვაზე. ჩვეულებრივ ამ მოდულებს წარმოადგენს პორტების დრაივერები, რომლებიც უშუალოდ მართავს აპარატურული საშუალებების მუშაობას.

გარდა ამისა, შეტანა/გამოტანის ქვესისტემა უზრუნველყოფს ფაილური სისტემის დრაივერების გარკვეულ სერვისს, რაც მათ კონკრეტული მოწყობილობებისადმი წაყენებული მოთხოვნების დამაკმაყოფილების საშუალებას აძლევს.

შეტანა/გამოტანის ქვესისტემა მუდმივად უნდა არსებობდეს მეხსიერებაში და მართოს მოწყობილობათა დრაივერების იერარქიის ერთობლივი მუშაობა. ამ იერარქიაში შეიძლება შედიოდეს გარკვეული ტიპის მოწყობილობების (ხისტი დისკების ან ლენტური დამგროვებლების) დრაივერები; მომწოდებლების მიერ მხარდაჭერილი დრაივერები (ისინი წაიტაცებს ბლოკური მოწყობილობებისადმი წაყენებულ მოთხოვნებს და მათ შეუძლია ამ მოწყობილობის არსებული დრაივერის ქცევის ნაწილობრივ შეცვლა, მაგალითად, მონაცემების დაშიფვრა); პორტების დრაივერები, რომლებიც მართავს კონკრეტული ადაპტერებს.

ის გარემოება, რომ დიდია ფაილურ სისტემიის დონეთა რაოდენობა, მოწყობილობათა დრაივეების ავტორებს მოქნილად მუშაობის საშუალებას აძლევს: დრაივერს შეუძლია მართვა თავის თავზე აიღოს მოთხოვნის შესრულების ნებისმიერ ეტაპზე (მოთხოვნის რეალიზების დაწყების მომენტიდან იმ მომენტამდე არსებულ მონაკვეთში, როდესაც ყველაზე დაბალ დონეზე ფუნქციონირებადი დრაივერი დაიწყებს კონტროლერის რეგისტრების დათვალიერებას). ფაილური სისტემის მუშაობის მრავალდონიანი მექანიზმი რეალიზებულია ამ დონეებზე გადასვლათა მწკრივების მეშვეობით.

ინიციალიზაციის მსვლელობისას მოწყობილობის დრაივერს შეუძლია ჩაერთოს ზემოთ აღნიშნულ მწკრივში და შემდეგ თვითონ განსაზღვროს დონე, რომელზეც საჭიროა მოხდეს გადასვლა. ეს ნიშნავს, რომ დრაივერი თავად ირჩევს მომდევნო დონეს, რაც მას, საჭიროებისამებრ ადრე შესრულებული ყველა ფუნქციის განმეორებით შესრულების საშუალებასაც აძლევს.

გამოძახების შესრულების მწკრივში შესული დრაივერის მიერ ინიცირებულ პროცედურას შეუძლია ეს მოთხოვნა შესაძლებლობის შემთხვევაში თვითონვე დააკმაყოფილოს, ან, საწინააღმდეგო შემთხვევაში, იგი დასაკმაყოფილებლად (შეუცვლელად ან შეცვლილი სახით) გადასცეს სხვა დონეზე.

III თავი
პროგრამულ უზრუნველყოფათა გალერეა

3.1. ოპერაციულ სისტემათა სახეები

ოპერაციული სისტემების განხილვა დავიწყოთ, დისკური ოპერაციული ***DOS*** სისტემით.

1. დისკური ოპერაციული *DOS*სისტემა

DOS(***D****isk* ***O****perating* ***S****ystem*)სისტემა ფართო გაგებით ინფორმაციის დისკურ დამგროვებლებზე (ხისტ დისკებსა და დისკეტებზე) ორიენტირებული კომპიუტერებისათვის განკუთვნილ ოპერაციული სისტემაა. კომპიუტერების ადრეულ ვერსიებს არ გააჩნდა დისკური დამგროვებლები და მათ მართავდა უდისკო ოპერაციული სისტემები. ასეთი კომპიუტერების მთავარი ნაკლი იყო ის, რომ მათში სამომხმარებლო პროგრამები ჩაიტვირთებოდა ლენტიდან, პერფობარათებიდან, კლავიატურიდან და ზღუდარების დაყენების გზით. მაგნიტური ტიპის დისკური დამგროვებლების გამოჩენის შემდეგ მათი მმართველი ქვეპროგრამების დამუშავება გახდა აუცილებელი. ოპერაციული სისტემისა და დისკური ინტერფეისის გაერთიანების შედეგად ფორმირებული იქნა ***დისკური ოპერაციული სისტემა***.

მაგნიტური ტიპის დისკურ დამგროვებლებთან შეღწევის გარდა დისკური ოპერაციული სისტემები ჩვეულებრივ სისტემურ ფუნქციებსაც ასრულებს. არსებობდა ***1960-1980*** წლებში *IBM*-ის მიერ გამოშვებული დიდი ელექტრონული გამომთვლელი მანქანებისა და მათი კლონებისათვის დამუშავებული ანალოგური დასახელების ოპერაციული სისტემები, მაგრამ ვინაიდან ფართოდ გავრცელდა მხოლოდ პერსონალური კომპიუტერებისათვის შექმნილი ***ერთამოცანური ტექსტური*** *MS-DOS* სისტემა, ამიტომ ვიწრო გაგებით ***„DOS“*** ტერმინის წორედ ამ სისტემის აღსანიშნავად გამოიყენება.

MS-DOS (***M****icrosoft* ***DOS***) *x86* არქიტექტურის ბაზაზე შექმნილი კომპიუტერებისათვის განკუთვნილი დისკური ოპერაციული სისტე-

მაა, რომელიც ***1980***-იანი წლებიდან ***1990***-იანი წლების ბოლომდე ყველაზე ხშირად გამოიყენებოდა ***IBM PC***-თავსებად კომპიუტერებში.

MS-DOS ფაქტობრივად ოპერაციული *CP/M* და*Unix* სისტემების არაოფიციალური ჰიბრიდია. პირველისგან მას მემკვიდრეობით ერგო სიმარტივე და ზერელობა, ხოლო მეორესგან – ფუნქციონალის უმეტესი ნაწილი. იგი მოქნილი ოპერაციული სისტემაა, რომელიც პერსონალური კომპიუტერის დიდ რესურსებს არ მოითხოვს.

1981 წელს *IBM*კორპორაციას დასჭირდა *IBM PC* კომპიუტერისათვის დასჭირდა ორიგინალური ოპერაციული სისტემა. მის შესაქმნელად ***Microsoft***-მა *Seattle Computer Products*-საგან შეიძინა უფლებები *86-DOS* სისტემაზე. *IBM*-ის მოთხოვნების შესაბამისად მისი მოდიფიცირების შედეგად შეიქმნა ახალისისტემა, რომელიც ***1981*** წლის აგვისტოში ბაზარზე გამოჩნდა *PC DOS* ***1.0*** ვერსიის სახით. შემდგომში ***1993*** წლამდე ***MS-DOS*** და ***PC DOS*** სისტემების დახვეწაზე ერთობლივად მუშაობდა ***Microsoft*** და ***IBM*** ფირმები. ***2000*** წლამდე, როდესაც ***Microsoft***–მა ***MS-DOS*** სისტემაზე საბოლოოდ აიღო ხელი, სულ გამოვიდა გამოვიდა ამ სისტემის რვა სრული და ორი ნაწილობრივი რელიზი. აღსანიშნავია, რომ ***MS-DOS*** საბაზისო შრედ იყო გამოყენებული *Windows*-ის ადრეული ვერსიებში. ***MS-DOS***-ის ბოლო დამოუკიდებელი ვერსია იყო ***6.22,*** მაგრამ ჩამტვირთველის სახით იგი მაინც განაგრძობდა არსებობას *Windows 95, Windows 98* და *Windows ME* სისტემებში.

2. OS/2 ოპერაციული ***OS/2*** სისტემა (***Operation Sistem/2***)წარმოადგენს მრავალამოცანურ, ერთმომხმარებლურ ოპერაციულ სისტემას, რომელიც მომხმარებელს უზრუნველყოფდა როგორც ტექსტური, ისე გრაფიკული ინტერფეისით. იგი ***IBM*** კორპორაციამ ***1987*** წელს ***Microsoft***–თან ერთად დაამუშავა. ამავე წელს ***IBM***-მა განაცხადა კომპიუტერულ ბაზარზე კომპიუტერების ***PS/2*** (***PersonalSystem/2***) ოჯახის გამოსვლის შესახებ.

ოპერაციულმა ***OS/2*** სისტემამ გამოჩენისთანავე გაიჩინა მრავალი თაყვანისმცემელი, განსაკუთრებით მაღალკვალიფიცირებული მომხმარებელთა წრიდან. იდეოლოგიით იგი ***Windows***-ის ოჯახის ოპერაციულ სისტემებს ჰგავს, თუმცა არ გააჩნია ამ უკანასკნელისათვის დამახასიათებელი ზოგიერთი მოხდენილი სამუშაო საშუალებები. ამუშავებისას, ისე როგორც ***Windows***-ის დროს, ეკრანზე ჩნდება სის-

ტემის *„სამუშაომაგიდა“*; მასზე განთავსებულია ობიექტებისიკონები, რომლებზედაც მაუსის მარცხენა ღილაკზე ორჯერ დაწკაპუნებით სამუშაოდ გაიხსნება აღნიშნული ობიექტები.

OS/2-საქვს ***DOS***-ის მძლავრი მხარდაჭერა, ამიტომ იგი ***DOS***-ისთვის დამუშავებული ყველა ***16***-ბიტური გამოყენებთან შეთავსებადია. ოპერაციული ***OS/2*** სისტემის მნიშვნელოვანი თავისებურებაა მაღალმწარმოებლური ფაილური ***HPFS*** (*„**H**igh **P**erformanse **F**ile**S**ystem“* - მაღალმწარმოებლური ფაილური სისტემა) სისტემის არსებობა. მონაცემთა ბაზების სერვერებზე ***DOS***-ის ფაილური სისტემის გამოყენების ნაცვლად უმჯობესია ***HPFS***-ის გამოყენება (***HPFS***-ს აქვს გრძელი სახელებიანი ფაილები).

1996 წლის სექტემბერში გამოსული ***OS/2***-ი მორიგი ***OS/2 Warp 4.0***ვერსია წარმოადგენდა აღნიშნული ოპერატიული სისტემის პიკსაც და „მზის ჩასვენების“ დასაწყისსაც.

OS/2-ის„ცხოვრებისეული გზა“ ერთ-ერთი ნათელი მაგალითია იმისა, რომ ნებისმიერი კარგად მოფიქრებული და ასევე ტექნიკურად კარგად რეალიზებული იდეა როგორ შეიძლება ჩაკლას დიდი ფულისკენ სწრაფვამ და ცუდად მოფიქრებულმა მენეჯმენტმა. პროექტი, რომელიც რეალიზებული ორიგინალური იდეებითა და არქიტექტურული გადაწყვეტებით რამდენიმე ნაბიჯით უსწრებდა კონკურენტებსა და ანალოგებს, აგრეთვე რომლის დამუშავებაზე დაიხარჯა მნიშვნელოვანი რესურსები და ადამიანური შრომა, ფაქტობრივად უდიდესი მოგების მიღებაზე კონკურენტული ბრძოლის დროს იქნა მოშთობილი. ***2004*** წელს ***IBM***-მა ხელი აიღო ***OS/2***-ის მხარდაჭერაზე, ხოლო მასზე უფლების მყიდველ ***Serenity Systems***-ს არ გააჩნია სათანადო ძალა პროდუქტის დასაწინაურებლად. ასეთი დასასრული არ ამცირებს იმ წვლილს, რომელიც აღნიშნულმა სისტემამ შეიტანა ოპერაციული სისტემების შექმნაში.

3. Unix

1965 წელს ამერიკული ტრანსნაციონალური საკომუნიკაციო ***AT&T*** კონგლომერატის ***Bell Telephone Laboratories*** ქვედანაყოფში მრავალდარგობრივ ***General Electric Company*** კორპორაციასა და ***მასაჩუსეტის ტექნოლოგიურ ინსტიტუტთან*** ერთად დაიწყეს მუშაობა ***Multics***(***Multi****plexed* ***I****nformation and* ***C****omputing* ***S****ervice*) სახელწოდების ისეთი ახალი მრავალამოცანური

ოპერაცული სისტემის შექმნაზე, რომელსაც შეეძლებოდა უზრუნველეყო რამდენიმე ასეული მომხმარებლის მომსახურება. სამუშაო ვერ დამთავრდა, რადგან ***1969*** წელს **Bell Labs** გავიდა პროექტიდან. მიუხედავად ამისა, **Bell Labs**-ის პროექტში მონაწილე თანამშრომლებმა ***კენტ ტომპსონმა*** (*Ken Tompson*) და ***დენის რიტჩიმ*** (*Dennis Ritchie*) სხვა თანამოაზრეებთან ერთად განაგრძეს მუშაობა დაპროგრამების მოსახერხებელი გარემოს შექმნაზე.

გამოიყენეს რა **Multics**-ზე მუშაობის გაჩენილი იდეები, ***1969*** წელს მათ შექმნეს მცირე ოპერაციული სისტემა, რომელსაც ჯგუფის წევრი ***ბრაინ კერნიგანის*** (*Brian Kernigan*) წინადადებით უწოდეს **Unix.**სისტემა საბოლოოდ დაიწერა და გამოიყენეს **PDP-7** ტიპის კომპიუტერზე. ***1971*** წელს **Bell Labs**-ში იგი გადაიტანეს უფრო მძლავრ **PDP-11** კომპიუტერში.

Unix-ის პირველი ასემბლერული ვერსიის შექმნის შემდეგ ***ტომპსონმა*** დაიწყო მუშაობა კომპილატორ **FORTRAN**-ის შექმნაზე, რომლის შედეგადაც მან დაამუშავა დაპროგრამების**B** სახელწოდების ენა. ამ უკანასკნელის გადამუშავებით ***რიტჩიმ*** შექმნა დაპროგრამების **C**სახელწოდების ენა, რომელიც სამანქანო კოდის გენერირების საშუალებას იძლეოდა.

1973 წელს**Unix**-ის ბირთვი გადაიწერა **C**ენაზე. განსაზღვრული მანქანისათვის მთლიანად ასემბლერის ენაზე დაწერილი ყველა წინანდელი ოპერაციული სისტემებისაგან განსხვავებით მოცემულ შემთხვევაში **UNIX**-ის ბირთვს ჰქონდა ასემბლერზე შედგენილი კოდის მხოლოდ ***10%*** (***1000*** სტრიქონი). ნებისმიერ მანქანაზე რომ ემუშავა, ახალ ოპერაციულ სისტემას სჭირდებოდა ასემბლერის ენაზე დაწერილი მხოლოდ რამდენიმე გვერდი და **C** ენის კომპილატორი. ეს საშუალებას იძლეოდა, რამდენიმე თვეში გადაგვეტანა ოპერაციული სისტემა სხვა აპარატურულ პლატფორმაზე და მასში საკმაოდ ადვილად შეგვეტანა ყველა სერიოზული ცვლილება და დამატება.

შემდგომში განუწყვეტლად იზრდებოდა **UNIX**–ის პოპულარობა და ***1975*** წელს გამოვიდა **Bell Labs**-ის გარეთ დამუშავებული მისი პირველი ვერსია. ოპერაციული სისტემისათვის **C** ენაზე დაწერილი საწყისი ტექსტების საყოველთაოდ ხელმისაწვდომობის გამო მრავალი კომპანია შეუდგა საკუთარი კომპიუტერისათვის **UNIX**–ის ადაპტირებას და გაჩნდა ამ ოპერაციული სისტემის უამრავი სხვადასხვა

ვერსია. ამან კიდევ უფრო გაზარდა ***UNIX***–ის პოპულარობა და ***1977*** წლიდან დაწყებული იგი გადაიტანეს სხვადასხვა აპარატურულ პლატფორმებზე. პარალელურად ხდებოდა სისტემის სრულყოფა, ფართოვდებოდა მისი შესაძლებლობები და იზრდებოდა მოდიფიკაციათა რაოდენობა.

UNIXგახდა ***პირველი გადასატანი ოპერაციული სისტემა*** და ეს იყო მისი წარმატების ერთ-ერთი მიზეზი. ***UNIX***–ის როგორც ადრეულ, ისე თანამედროვე ვერსიებში მუდმივად შეიტანებოდა და შეიტანება ცვლილებები. ეს, ***ერთი მხრივ,*** აფართოებს სისტემის შესაძლებლობებს, ამაღლებს მის სიმძლავრესა და საიმედოობას, ხოლო ***მეორე მხრივ,*** აღრმავებს განსხვავებებს არსებულ ვერსიებს შორის. ამიტომ წამოიჭრა ***სისტემის სხვადასხვა თვისების სტანდარტიზაციის აუცილებლობა***. სტანდარტების არსებობა აიოლებს გამოყენებათა გადატანას ***UNIX***–ის სხვადასხვა ვერსიებს შორის და იცავს როგორც მომხმარებლებს, ისე პროგრამული უზრუნველყოფის მწარმოებლებს. ამიტომ ***1980***-იან წლებში წარმოიშვა და დამუშავდა მთელი რიგი სტანდარტი, რომელმაც გავლენა მოახდინა ***UNIX***–ის განვითარებაზე.

დღეს ***UNIX***სახელწოდებაში სხვადახვა მწარმოებლების მიერ საკუთარი კომპიუტერისათვის დამუშავებული ათობითი ვერსიაა გაერთიანებული. ***UNIX***–ის პოპულარობის მიზეზებია:

■ სისტემის კოდი დაწერილიადაპროგრამების მაღალი დონის ***C*** ენაზე, რის გამოც ადვილია მისი გაგება, შეცვლა და სხვა პლატფორმაზე გადატანა. შეიძლება ვთქვათ, რომ ***UNIX ერთ-ერთი ყველაზე ღია სისტემაა***;

■ ***UNIX*** მრავალამოცანური და მრავალსამომხმარებლო სისტე-მაა. ერთ მძლავრ სერვერს შეუძლია მრავალ მომხმარებელს მოემსახუროს. ამ დროს მხოლოდ ერთი სისტემის ადმინისტრირებაა აუცილებელი. გარდა ამისა, სისტემას შეუძლია დიდი რაოდენობის სხვადახვა ფუნქცია შეასრულოს; კერძოდ, მას შეუძლია იმუშაოს გამომთვლელ სერვერად, მონაცემთა ბაზის სერვერად, ქსელურ სერვერად, მხარი დაუჭიროს ქსელის უმნიშვნელოვანეს სერვისებს და ა. შ.;

■ ***UNIX***–ის ვერსიების მრავალფეროვნების მიუხედავად ყველა მათგანს აქვს ერთნაირი არქიტექტურა და სტანდარტული ინტერფეისე-

ბი. ამიტომ ადმინისტრატორი ადვილად გადადის სხვა ვერსიაზე, ხოლო მომხმარებლებისათვის ვერსიების ცვლა შეუმჩნეველია;

- ***მარტივი, მაგრამ მძლავრი მოდულური სამომხმარებლო ინტერფეისი.*** ვიწრო სპეციალიზებული ამოცანების გადამწყვეტ უტილიტების მეშვეობით შეგვიძლია ავაგოთ რთული კომპლექსები.
- ***ფაილური სისტემის უნიფიცირებული ინტერფეისის არსებობა,*** რაც საშუალებას გვაძლევს შევაღწიოთ როგორც დისკზე შენახულ მონაცემებში, ისე ტერმინალებში, პრინტერებში, ქსელში და ა. შ.;
- უმარტივესი ტექტური რედაქტორებიდან დაწყებული, მონაცემთა ბაზების მმართველი მძლავრი სისტემებით დამთავრებული, უამრავი, (მათ შორის თავისუფლად განვრცელებადი) გამოყენებათა არსებობა.

4. Linux

ყველა თანამედროვე ***UNIX***-სისტემებისათვის დამახასიათებელია ის გარემოება, რომ თითოეული მათგანი დიდი მოცულობისაა და რთულია. ეს გარკვეულწილად წინააღმდეგობაში მოდის თავად ***UNIX***-ში ჩადებულ ორიგინალურ იდეასთან, რომელიც მიმართული იყო მოცულობის შემცირებისა და თვალსაჩინოების ამაღლებისაკენ. ყველა საწყისი კოდიც რომ იყოს თავისუფლად ხელმისაწვდომი (რაც უმეტეს შემთხვევაში ასე არ არის), მათი გაგება მაინც ცდება ერთი ადამიანის შესაძლებლობებს. ამიტომ წინა საუკუნის ***80***-იან წლებში აქტუალური გახდა ***UNIX***-ის მსგავსი ახალი სისტემების დაწერა. ***1987*** წელს ამსტერდამის თავისუფალი უნივერსიტეტის პროფესორმა, ამერიკელმა მეცნიერმა ***ენდრიუ ტანენბაუმმა*** (*Andrew Tanenbaum, 1944*) შექმნა ***MINIX*** სახელწოდების საკმაოდ მომცრო ***სასწავლო სისტემა,*** რომელიც შეიცავდა ***C*** ენაზე დაწერილ ***11800*** და ასემბლერის ენაზე დაწერილ ***800*** სტრიქონებს. ამ სისტემის ინტერნეტში განსჯის პროცესში მრავალი სპეციალისტი სთხოვდა ***ტანენბაუმს*** გაეფართოვებინა სისტემის ფუნქციური შესაძლებლობები. ავტორმა ამ თხოვნებზე უარით უპასუხა, რათა არ გაზრდილიყო სისტემის ზომა, რაც საშუალებას აძლევდა სტუდენტებს აღნიშნული სისტემა ერთი სემესტრის განმავლობაში ათვისებინა. უარყოფითი პასუხები მრავალ მომხმარე-ბელს აღიზიანებდა და ბოლოს და ბოლოს ფინელმა სტუდენტმა ***ლინუს ტორვალდსმა*** (*Linus Torvalds*) გადაწყვიტა შეექმნა ***UNIX***-ის კიდევ ერთი კლონი, რომელსაც მან ***Linux*** უწოდა. იგი უნდა ყოფილიყო არა სასწავლო, არა-

მედ სრულყოფილი საწარმოო სისტემა, რომელშიც ჩადებული იქნებოდა ***MINIX***-ში არარსებული ფუნქციები. მისი პირველი ***0.01*** ვერსია ***1991*** წელს გამოვიდა. ***MINIX***-საგან განსხვავებით იგი იყო ***მონოლითური სისტემა***, ე.ი. მთელი ოპერაციული სისტემა ბირთვში იყო განთავსებული.საწყისი ტექსტი შეიცავდა ***C*** ენაზე დაწერილ ***9300*** და ასემბლერის ენაზე დაწერილ ***950*** სტრიქონებს. ფაქტობრივად იგი იყო ***გადაკეთებული MINIX*** – ერთადერთი სისტემა, რომლის საწყისი კოდი ჰქონდა ***ტორვალდსს***. აღნიშნული სისტემის სრულყოფის პროცესში ჩაერთო ათასობით სპეციალისტი, დამუშავდა მისი უამრავი ვერსია და ამგვარად ჩამოყალიბდა ოპერაციულ სისტემათა ***Linux*** სახელწოდების ოჯახი. მასში შემავალი სისტემები ერთადერთი უფასო სისტემებია, რომლებმაც ძალიან დიდი პოპულარობა მოიპოვა. მსოფლიოში გამოთვლითი ტექნიკის ყველაზე რეიტინგულ პროდუქტების *TOP500* ჩამონათვალის მიხედვით ***Linux*** გამოიყენება მძლავრი სუპერკომპიუტერების ***97%***-სა და სერვერების ***60%***-ში. ***Linux***–ის განვითარების *Linux Foundation* ცენტრის თანახმად ***მონაცემების დამამუშავებელ ცენტრებსა*** ***(datacenter***-*ებსა*)და საწარმოებში ჩაშენებული სისტემების ბაზრის ნახევარი ***Linux***-ს უკავია; ***2009*** წლის მონაცემებით იგი გამოიყენება ***ნოუთბუკების 32%***-ში, ხოლო პერსონალური კომპიუტერებში გამოყენების მხრივ იგი სტაბილურად მესამე ადგილზეა (გამოიყენება ამ კომპიუტერების ***1***-დან ***3%***-ში) .მსოფლიოში უმსხვილესი საინვესტიციო ბანკ *GoldmanSachs*-ის გამოკვლევის მიხედვით ელექტრონულ მოწყობილობათა ბაზრის ***42%Linux***-ს უკავია.

5. Mac OS

Mac OS (***Macintosh Operating System***) წარმოადგენს კორპორაცია *Apple*-ის მიერ ***1984***წელს პერსონალური კომპიუტერ *Macintosh*-ის მწკრივისათვის დამუშავებულ პროპრიეტარული (***proprietary*** – კერძო, დაპატენტებული, საკუთარი. ***პროპრიეტარული პროგრამული უზრუნველყოფა*** შეიძლება იყოს როგორც კომერციული, ისე უფასოც) ოპერაციული სისტემების ოჯახს. ***1990*** წლიდან ***Mac OS*** ტერმინი ოფიციალურად გამოიყენება *Macintosh*-ისათვის დამუშავებული ყველა ოპერაციული სისტემის აღსანიშნავად.

Apple-კომპანიის წარუმატებელი მარკეტინგული პოლიტიკის გამო *Macintosh* ტიპის კომპიუტერები მასობრივი ბაზრისათვის ბრძო-

ლაში დაამარცხა ***IBM PC***-თავსებადმა კომპიუტერებმა და დღეს მას კომპიუტერების მსოფლი პარკის მხოლოდ ***3%*** უკავია.

საიმედოობითა და მოხერხებულობებით ***Mac OS*** ჯობნის ***Windows***-ს. თავიდანვე იყო ჩაფიქრებული, რომ იგი ძალიან მეგობრული ყოფილიყო მომხმარებლისადმი და ადვილი ყოფილიყო მისი გამოყენება დაკაბადონებისა და პოლიგრაფიის პროფესიულ სისტემებში. ამიტომ მძლავრ საგამომცემლო სისტემებისათვის იგი შეუცვლელია.

6. Windows.

დღეისთვის ყველაზე მეტად გავრცელებულია კორპორაცია ***Microsoft***-ის მიერ დამუშავებული ***Windows***-ის ოჯახში შემავალი გრაფიკული ოპერაციული სისტემები.

1985 წელს ოპერაციულ სისტემა ***DOS***-ისათვის ***Microsoft***-მა დაამუშავა პირველი გრაფიკული ***გარსაცმი Microsoft Windows***. ამერიკელები მას დღემდე თვლიან ***XX*** საუკუნის ერთ-ერთ უდიდეს გამოგონებად.

გარსაცმი(***shell***) *ოპერაციული სისტემის ზედნაშენია, რომელიც მნიშვნელოვნად ამარტივებს მომხმარებლის მუშაობას. იგი არის ოპერაციული სისტემის ბრძანებათა ინტერპრეტატორი, რომელიც ცალკეულ ბრძანებებს მიღებისთანავე აანალიზებს და ასრულებს.*

1992 წლის აპრილში გამოჩნდა ახალი ტექნოლოგიის (***New Technology***) მრავალსამომხმარებლო ოპერაციული ***Windows NT*** სისტემა. იგი წარმოადგენდა ***MS DOS*** სისტემით მართულ ოპერაციულ სისტემას, რომელიც შექმნილი იყო მაღალმწარმოებლური სერვერებისა და მუშა სადგურებისათვის. მის მნიშვნელოვან სიახლეს წარმოადგენდა ახალი ფაილური სისტემა ***NTFS*** (***NT File System***), რომელიც მაღალ დონეზე აკონტროლებდა რესურსებთან შეღწევას და რომელსაც შეეძლო ამოვარდნათა შედეგების ლიკვიდირება.

1995 წლის ***24*** აგვისტოს *Microsoft*–მა გამოუშვა ***Windows 95*** სისტემა. იგი უკვე იყო არა ***MS DOS***–თვის შექმნილი გრაფიკული გარსაცმი, არამედ *Microsoft*–ის ოჯახში შემავალი ***პირველი სრულფასოვანი 32-ბიტური ოპერაციული სისტემა,*** რომელსაც შეეძლო ***MS DOS***–თვის შექმნილ ***16***-ბიტურ გამოყენებებთან მუშაობაც.

1998 წლის ***25*** ივნისს გამოჩნდა ***Windows***–ის ოჯახის მომდევნო წევრი ***Windows 98***. იგი წარმოადგენდა ***Windows 95***–ის ლოგიკურ გაგრძელებას, რომელსაც ჰქონდა უფრო მაღალი მწარმოებლურობა და გაზრდილი მულტიმედიური შესაძლებლობები.

1999 წლის ***15*** დეკემბერს დაიწყო მრავალპროცესორული დამუშავებისა და ინფორმაციის დაცვის საშუალებებით აღჭურვილი მომდევნო თაობის ქსელური ოპერაციული სისტემა ***Windows 2000 Server***-ის წარმოება, რომელსაც ***2016*** წლის ივლისის შემდეგ პრაქტიკულად არავინ არ იყენებს.

2000 წლის ***14*** სექტემბერს გამოვიდა ***Windows ME*** (***M****illennnium* ***E****dition* – საუკუნის ნაკეთობა) სისტემა. ***Windows 98***–თან შედარებით მას ჰქონდა გაფართოებული მულტიმედიური შესაძლებლობები, ქსელში შეღწევის გაუმჯობესებული საშუალებლოები, გაუმჯობესებული საცნობარო სისტემა და იყენებდა უახლოეს მოწყობილობებს.

2001 წლის ***25*** ოქტომბერს *Microsoft*–მა გამოუშვა ***Windows ME***-სა და ***Windows 2000***-ის შერწყმის შედეგად მიღებული ოპერაციული სისტემა ***Windows XP.*** აღნიშნული ინტეგრაციის შედეგად მიღებული იქნა ერთ-ერთი საუკეთესო ოპერაციული სისტემა. იგი აღჭურვილი იყო გაუმჯობესებული სამომხმარებლო ინტერფეისით, რომელიც მაქსიმალურად აადვილებდა სხვადასხვა მიზნით კომპიუტერის გამოყენებას.

2006 წლის ***30*** ნოემბერს კორპორატიული, ხოლო ***2007*** წლის ***30*** იანვრისათვის – ჩვეულებრივი მომხმარებლებისათვის გამოვიდა ოპერაციული სისტემა ***WindowsVista,*** რომელმაც მასში არსებული მთელი რიგი ნაკლოვანებების გამო მომხმარებელთა სიყვარული ვერ დაიმსახურა. ***2009*** წლის ***22*** ოქტომბერს იგი შეცვალა ოპერაციულმა ***Windows7*** სისტემამ, რომელშიც აღმოფხვრილიიყო***Windows Vista***-სთვის დამახასიათებელი ნაკლოვანებები.

2012 წლის ***26*** ოქტომბერს დაიწყო ოპერაციული სისტემა ***Windows8***-ის გაყიდვა, მაგრამ ***2016*** წლის ***12*** იანვრიდან ***Microsoft***–მა შეწყვიტა მისი მხარდაჭერა.

2014 *წლის* სექტემბერში მომხმარებლებს წარუდგინეს ოპერაციული სისტემა ***Windows10,*** რომელიც ბაზარზე გამოჩნდა ***2015*** წლიდან. მის დასახელებაში გამოტოვეს რიცხვი ***9*** და გამოიყენეს რიცხვი ***10.***

სოციოლოგიური გამოკვლევების თანახმად, აღნიშნული ვერსია ***Windows***-ის ვერსიებს შორის ყველაზე სკანდალური ვერსიაა. მკვლევრების ***50%*** მას მავნე პროგრამულ უზრუნველყოფად თვლის. ამას განაპირობებს ის გარემოება, რომ *Microsoft*–მა მასში ჩააშენა თვალთვალის უპრეცედენ-

ტო რაოდენობის მოდულები. ხელში შეიძლება ჩაგდებული იქნეს ყველაფერი – საფოსტო მიმოწერიდან დაწყებული კლავიშებზე თითის დაჭერითა და პაროლის შეყვანით დამთავრებული. შეგროვილი ინფორმაცის *Microsoft*–ში გადაიგზავნება სამუდამოდ შესანახად. სალიცენზიო ხელშეკრულებაში *Microsoft*-ს ოფიციალურად ენიჭება მომხმარებელთა თვალთვალის უფლება.

***ცხრ. 3.1. Windows**–ის ვერსიების გავრცელებადობა სხვადასხვა წყაროების მიხედვით* (https://ru.wikipedia.org/wiki/Windows)

წყარო / ვერსია	«Net Market Share», 02. 2016	«StatOStats.ru», 2. 2016	«Net Applications», 04. 2016	«StatCounter», 04. 2016
ყველა ვერსია	88,66 %	90,10 %	88,77 %	83,29 %
Windows 10	12,82 %	29,66 %	15,34 %	18,88 %
Windows 8	12,26 %	20,26 %	13,04 %	13,37 %
Windows 7	52,34%	32,07%	47,82%	43,95%
WindowsXP	11,24%	7,42%	10,63%	7,09%

Windows 10-ის ***მეორე პრობლემა*** თვითნებობაა. ოპერაციულ სისტემას შე-უძლია მომხმარებლის კომპიუტერიდან წაშალოს ნებისმიერი პროგრამა, მომხმარებლის უკითხავად ჩართოს მიკროფონი, ან კომპიუტერში დააყენოს ისეთი პროგრამა, რომლის დაყენება მომხმარებელს არ სურს. ***მესამე პრობლემაა*** – პროგრამულიკოდის დაბალი ხარისხი. დაფიქსირებულია კომპიუტერიდან ინფორმაციის დაკარგვის მრავალი შემთხვევა და მორიგი განახლებით გამოწვეული დაზიანებები. მოცემულ მომენტში ***Windows***–ის ყველაზე გავრცელებულ ვერსიას წარმოადგენს ***Windows 7*** (ცხრ. 3.1).

7. სხვადასხვა სახის კომპიუტერში გამოყენებული ოპერაციული სისტემების ზოგადი დახასიათება

კომპიუტერულ სამყაროში არსებობს მრავალი სახის ოპერაციული სისტემა. ზემოთ ინდივიდუალურად დავახასიათეთ მათი პოპულარული ოჯახები. მოცემულ პარაგრაფში ზოგადად დავახასიათებთ ზოგიერთ სპეციფიკური სახის გამოთვლითი ტექნიკაში გამოყენებულ ოპერაციულ სიტემებს, რომელთა ცოდნა გააფართოებს მკითხველის თვალსაწიერს.

ჩაშენებული ოპერაციული სისტემები. ჩაშენებული ოპერაციული სისტემები გამოიყენება სხვადასხვა მოწყობილობის მმართველ კომპიუტერებში. მათზე სამომხმარებლო პროგრამების დაყენება არაა გათვალისწინებული, ამიტომ ხშირად მათ კომპიუტერებადაც არ მიიჩნევენ. ჩაშენებულ კომპიუტერებს აყენებენ ტელევიზორებზე, ავტომობილებზე, მიკრო-

ტალღოვან ღუმლებზე და ა.შ. ჩაშენებული სისტემები ვერ იყენებს გარე პროგრამული უზრუნველყოფას. მიკროტალღოვან ღუმელში შეუძლებელია ჩავტვირთოთ ახალი პროგრამა, რამდენადაც ყველა მისი პროგრამა მუდმივ დამხსომებელ მოწყობილობაშია ჩაწერილი. ამიტომ საჭირო არ არის პროგრამები დავიცვათ ურთიერთზემოქმედებებისაგან, რაც ამარტივებს ოპერაციულ სისტემას. ამ სფეროში გავრცელებული ოპერაციული სისტემებია ***QNX*** და ***VxWorks***.

<u>სენსორული კვანძების ოპერაციული სისტემები.</u>

ხშირად საჭიროა ***მინიატურული სენსორული კვანძებისავან*** ავაგოთ ერთმანეთთან და საბაზისო სადგურთან უმავთულო არხებით დაკავშირებული ქსელები. ისინი გამოიყენება, მაგალითად, ნაგებობათა პერიმეტერებისა და სახელმწიფო საზღვრების დასაცავად, გარემოს ტემპერატურისა და ნალექების დონეთა გასაზომად, საბრძოლო ველზე მოწინააღმდეგის გადაადგილების შესახებ ინფორმაციის შესაგროვებლად და ა. შ.

თითოეული ***სენსორული კვანძი*** პროცესორით, ოპერატიული მეხსიერებით, მუდმივი დამხსომებელი მოწყობილობითა და ერთი ან რამდენიმე გადამწოდით აღჭურვილი ***ნამცილი კომპიუტერია.*** მასზე მუშაობს ასევე ნამდვილი ***ოპერაციული სისტემა***, რომელიც ჩვეულებრივ შექმნილი მოვლენებით იმართება – იგი პასუხობს აღნიშნულ გარე მოვლენებს ან ჩაშენებული საათების სიგნალების მიხედვით პერიოდულად ასრულებს გარკვეულ გაზომვებს. ოპერაციული სისტემა მცირე მოცულობის და მარტივი უნდა იყოს, რადგან მცირეა ამ კვანძების ოპერატიული მეხსიერების ტევადობა და ბატარეის მუშაობის ხანგრძლივობა. ჩაშენებული სისტემების მსგავსად აღნიშნულ კვანძებში წინასწარაა ჩატვირთული ყველა პროგრამა და მომხმარებელს არ შეუძლია მოულოდნელად აამუშაოს ინტერნეტიდან ჩატვირთული პროგრამა, რაც მნიშვნელობნად ამარტივებს მთელ კონსტრუქციას. სენსორული კვანძების საყოველთაოდ ცნობილი ოპერაციული სისტემაა ***TinyOS.***

<u>რეალური დროის ოპერაციული სისტემები.</u>

რეალური დროის სისტემებისათვის დრო საკვანძო პარამეტრია. რეალური დროის რეჟიმში მომუშავე კომპიუტერებმა, მაგალითად, უნდა შეაგროვოს ცნობები პროცესების შესახებ და ისინი გამოიყენოს საწარმოოში ჩარხების მართვისათვის. საკმაოდ ხშირად ისინი უნდა შეესაბამებოდეს ძალიან ხისტ დროით მოთხოვნებს. მაგალითად, როდესაც ავტომობილი ამკრეფ კონვეიერზე გადანაცვლდება, მაშინ დროის გარკვეულ მომენტებში უნდა შესრულდეს სრულად კონკრეტული ოპერაციები. შემდულებელი რობოტი წინსწრებულად ან შეყოვნებით თუ დაიწყებს შედულებას, მაშინ მანქანა უვარგისი გახდება.

ერთმანეთისაგან განასხვავებენ ***ხისტი რეალური დროის*** და ***მოქნილი რეალური დროის*** სისტემებს. ***პირველ შემთხვევაში*** ოპერაცია დროის ზუს-

ტად მოცემულ მომენტში, ან დროის ზუსტად მოცემულ შუალედში უნდა შესრულდეს. ***მეორე შემთხვევაში*** ასეთი სიზუსტის დარღვევა თუმცა არაა სასურველი, მაგრამ იგი მაინც სრულად დასაშვებია. ***ხისტი რეალური დროის სისტემებს*** მიეკუთვნება, მაგალითად, საავიაციო-კოსმოსური ელექტრონული მოწყობილობათა საწარმოო პროცესების მართვის სისტემები, ასევე სამხედრო ან მსგავსი სფეროებში არსებული წარმოებების მართვის სისტემები; ***მოქნილი რეალური დროის სისტემათა*** კატეგორიას მიეკუთვნება ციფრული აუდიო- ან მულტიმედიური სისტემები.

რეალური დროის სისტემებს წაეყენება ძალიან ხისტი მოთხოვნები, ამიტომ ზოგჯერ ოპერაციული სისტემები წარმოადგენს გამოყენებით სისტემებთან შეუღლებულ მარტივ ბიბლიოთეკას, სადაც ყველაფერი მჭიდროდაა ურთიერთდაკავშირებული და ამ სისტემის ნაწილებს შორის რაიმე დაცვა არ არსებობს. ასეთ სისტემას, მაგალითად, წარმოადგენს ***e-Cos.***

სმარტბარათების ოპერაციული სისტემები. ყველაზე პატარა ოპერაციული სისტემები ***სმარტბარათებში***გამოიყენება. ისინი საკუთარი პროცესორის მქონე საკრედიტო ბარათის ზომის მოწყობილობებია. მასში გამოყენებულ ოპერაციულ სისტემას უნდა შეეძლოს მეტად შეზღუდული გამოთვლითი სიმძლავრის მქონე პროცესორსა და ასევე ძალიან მცირე მოცულობის მეხსიერებასთან მუშაობა. განასხვავებენ კონტაქტურ და უკონტაქტო სმარტბარათებს. ***კონტაქტური სმარტ ბარათები*** იკვებება წამკითხავი მოწყობილობასთან კონტაქტის მეშვეობით, ხოლო ***უკონტაქტო სმარტბარათები*** – ინდუქციის ეფექტის მეშვეობით, რაც მნიშნელოვნად ზღუდავს მათ შესაძლებლობებს. ზოგიერთ სმარტბარათებს შეუძლია ერთადერთი, მაგალითად, გადახდებთან დაკავშირებული ფუნქციის შესრულება, მაგრამ არსებობს მრავალფუნქციური სმარტბარათებიც. ხშირად ისინი მიეკუთვნება დაპატენტებულ სისტემებს.

ზოგიერთი სმარტბარათი იყენებს დაპროგრამების ენა ***Java***-ს. ამ შემთხვევაში სმარტბარათის მუდმივი დამხსომებელი მოწყობილობა შეიცავს ***Java Virtual Machine*** (***JVM – Java***-ს ვირტუალური მანქანა)ინტერპრეტატორს. ბარათში ჩაიტვირთება ***Java***-ინტერპრეტატორის მიერ შესრულებული ***Java***–**აპლეტები** (**მომცრო პროგრამები**). ზოგიერთ ბარათს შეუძლია ერთდროულად რამდენიმე ***Java***–აპლეტს გაართვას თავი, რისი წყალობითაც იგი მულტიდაპროგრამების რეჟიმში მუშაობს და პროგრამებს რიგითობის დაცვით ასრულებს. ორი და მეტი აპლეტის ერთდროულად შესრულებისას აქტუალური ხდება რესურსების მართვისა და დაცვის საკითხები, რომლებიც უნდა გადაწყვიტოს ბარათზე არსებულმა ოპერაციულმა სისტემამ, რომელიც, როგორც წესი, ძალიან პრიმიტულია.

3.2. ოპერაციულ *Windows* სისტემათა თავისებურებები

რამდენადმე სრულად განვიხილოთ დღეს ფართოდ გავცელებული ***Windows*** სახის ოპერაციული სისტემების აგებულებისა და ფუნქციონირების ზოგიერთი საკვანძო საკითხი.

1. ოპერაციული სისტემა Windows–ის არქიტექტურის თავისებურებები

ოპერაციული სისტემა ***Windows***-ის არქიტექტურას აქვს ***მოდულური სტრუქტურა,*** ე. ი. იგი აგებულია საკუთარი ფუნქციათა ხისტად შემსრულებელი დანაწევრებადი კომპონენტებისაგან, რომლებიც ორ დონედაა განაწილებული. ქვედა დონეზე განთავსებულია ე. წ. ***ბირთვის რეჟიმში*** (***user mode***)მომუშავე, ხოლო ზედა დონეზე – ***სამომხმარებლო რეჟიმში*** (***user mode***) მომუშავე კომპონენტები. მომხმარებელი სისტემურ პროგრამებსა და ოპერაციული სისტემის შინაგან კომპონენტებს ბრძანებებს სამომხმარებლო რეჟიმის მეშვეობით უგზავნის, ეს ბრძანებები კი ***ბირთვის რეჟიმში*** სრულდება. ეს ძალიან მნიშვნელოვანია, რადგან აღნიშნულ პროგრამებსა და კომპონენტებთან უშუალოდ მუშაობისას მომხმარებელს შეიძლებოდა ***შეცდომით*** შეეცვალა ან გაეძევებინა რომელიმე მათგანი, რაც გამოიწვევდა ოპერაციული სისტემის რღვევის პროვოცირებას. საკუთარი პრივილეგიები აქვს როგორც სამომხმარებლო, ისე ბირთვის რეჟიმს, მაგრამ უკანასკნელი უფრო მეტი პრივილეგიებით სარგებლობს.

■ ***<u>ბირთვის რეჟიმი</u>*** ოპერაციული სისტემის ძალიან მნიშვნელოვანი და განუყოფელი ნაწილია. ***ბირთვი*** (***Kernel***) - ოპერაციული სისტემის ცენტრალური ნაწილია, რომელიც მართავს პროცესებს, გამოთვლითი სისტემის რესურსებს და პროცესებს ამ რესურსებთან კოორდინირებულად შეღწევის საშუალებას აძლევს. იგი გარკვეული ***აბსტრაქციაა,*** რომლის მეშვეობითაც მომხმარებელი მოიხმარს კომპი-უტერთან მუშაობისათვის საჭირო რესურსებს.

ფიზიკურად ***ბირთვი*** ოპერაციული სისტემის კოდის მცირე ნაწილია და მიეკუთვნება სისტემის ყველაზე ინტენსიურად გამოყენებადი კომპონენტების რიცხვს. ***<u>პერსონალური კომპიუტერის ჩართვისას ოპერატიულ მეხსიერებაში გარე მეხსიერებიდან (დისკიდან) ოპერაციული სისტემის ბირთვი გადმოდის,</u>*** ხოლო მისი დანარჩენი ნაწი-

ლი დისკზე რჩება. ოპერატიულ მეხსიერებაში ისინი საჭიროებისამებრ გადაიტანება და საკუთარი ფუნქციების შესრულების შემდეგ ხელახლა ბრუნდება გარე მეხსიერებაში.

ბირთვის მუშაობა დაფარულია მომხმარებლისათვის: მას ამ მუშაობის პროცესში უშუალოდ ჩარევის უფლება არა აქვს. არსებობს ბირთვის რამდენიმე სახე: მონოლიური ბირთვი, მიკრობირთვი, ეკზობირთვი, ნანობირთვი და ჰიბრიდული ბირთვი. ჩვენ მხოლოდ ***მონოლითურ*** და ***მოდულურ ბირთვს*** განვიხილავთ, რადგან **Windows**–ში თავდაპირველად ***მონოლითური ბირთვი*** იქნა გამოყენებული, რომელმაც შემდეგ ***მოდულური ბირთვის*** სახე მიიღო.

მონოლითური ბირთვი ოპერაციულ სისტემებში გამოყენებული ბირთვების კლასიკური არქიტექტურაა. ისინი გვაწვდის მოწყობილობათა აბსტრაქციის მდიდარ ნაკრებს. მისი ყველა ნაწილი ერთ სამისამართო სივრცეში მუშაობს. ე. ი. კომპიუტერის მეხსიერებაში შეინახება ინფორმაცია, რომლის ყოველ მდგენელს შეესაბამება შენახვის ადგილის განმსაზღვრელი რიცხვი – მისამართი. ოპერაციული სისტემებში დღესაც ყველაზე ხშირად ***მონოლითური ბირთვები*** გამოიყენება. მათი განვითარების პროცესი დიდიხანია რაც გრძელდება. ამის გამო სწორედ ისინია არქიტექტურულად ყველაზე მეტად სრულყოფილი და საექსპლოატაციოდ მოსახერხებელი. მიუხედავად ამისა მას აქვს შემდეგი ორი ნაკლი:

1. კომპიუტერის აპარატურული უზრუნველყოფის შემადგენლობის შეცვლისას სრულად უნდა შეიცვალოს ბირთვის კომპილაცია, ე. ი. მოხდეს მისი გადაკომპილირება;

2. ერთ სამისამართო სივრცეში მუშაობის გამო რომელიმე კომპონენტის მოულოდნელმა ამოვარდნამ შეიძლება მთელი სისტემა მუშაობის უუნარო გახადოს.

ზემოთ აღნიშნული ნაკლოვანებების აღმოსაფხვრელად მოხდა მონოლითური ბირთვის მოდიფიცირება, რის შედაგადაც მიღებული იქნა ***მოდულური ბირთვი***. მაშასადამე, ეს უკანასკნელი ერთ სამისამართო სივრცეში მომუშავე ზემოთ აღნიშნული ნაკლოვანებებისაგან თავისუფალი ***მოდიფიცირებული მონოლითური ბირთვია***. კომპიუტერულ სისტემისათვის ახალი აპარატურის დამატებისა ბირთვის სრული გადაკომპილირების ნაცვლად მასში შეგვიძლია დინამიკურად (ბირთვის მუშაობის შეუწყვეტლად და კომპიუტერის გადაუტვირთველად)

შევტვირთოთ ამ აპარატურის მხარდამჭერი მოდული, მაგალითად, მისი დრაივერი. გარდა ამისა, მოდულური ბირთვის შემთხვევაში სისტემა თავისუფალია კომპონენტების ისეთი ამოვარდნებისაგან, რომელსაც შეუძლია მთალი სისტემა მუშაობის უუნარო გახადოს.

მოდულის შეტვირთვის შედეგად ბირთვის შესაძლებლობების გაფართოების მიუხედავად ***ბირთვი მაინც მონოლითურად რჩება*** და კვლავ ერთ სამისამართო სივრცეში განაგრძობს მუშაობას. მაშასადამე, არსებობს ორი სახის მონოლითური ბირთვი. პირველ მათგანში შეუძლებელია, ხოლო მეორე მათგანში – შესაძლებელია ახალი მოდულის შეტვირთვა. პირველ მათგანს ეწოდება ***კლასიკური-მონოლითური,*** ხოლო მეორე მათგანს – ***მოდულური*** ბირთვი.

ბირთვს მრავალი ფუნქციის შესრულება შეუძლია. იგი გადართავს კონტექსტებს, ახდენს გამოთვლის პროცესის ორგანიზებას, ჩატვირთავს და გადმოტვირთავს გვერდებს, ამუშავებს შეწყვეტებს.

მომხმარებლის რეჟიმში მომუშავე დანართებს, თავის მხრივ, შეუძლია საჭირო სამუშაოების შესასრულებლად მიმართოს ბირთვს. ***ბირთვის მიერ ფუნქციათა შესრულების სიჩქარე ოპერაციული სისტემის მწარმოებლურობის მაჩვენებელია.***

ბირთვი კომპიუტერული სისტემის მთავარი შემადგენელი ნაწილია, რომელიც მას მუშაობის საშუალებას აძლევს. იგი შედგება შემდეგი შრეებისაგან: ***1)*** აპარატურული მხარდაჭერის საშუალებათა შრე; ***2)*** სამანქანო-დამოკიდებული კომპონენტების შრე; ***3)*** ბირთვის საბაზისო მექანიზმების შრე; ***4)*** რესურსთა მენეჯერების შრე; ***5)***სისტემურ გამოძახებათა ინტერფეისის შრე.

ბირთვის რეჟიმში მომუშავე პროგრამულ უზრუნველყოფა: ***ა)*** პირდაპირ და შეუზღუდავად აღწევს აპარატურულ უზრუნველყოფაში; ***ბ)*** აღწევს კომპიუტერის მთელ მეხსიერებაში; ***გ)*** ვერ გაძევდება ხისტი დისკის ფაილში; ***დ)*** ფლობს სამომხმარებლო რეჟიმის პროცესებთან შედარებით უფრო მაღალ პრიორიტეტს;

სხვა პროგრამათა ზემოქმედებისაგან საკუთარი კომპონენტების დაცვის უნარი არქიტექტურულადაა უზრუნველყოფილი.

დავირუსებულ ბირთვს შეუძლია დაანგრიოს ოპერაციული სისტემა. ამიტომ ინტერნეტიდან დრაივერების ჩატვირთვისას ფრთხილად უნდა ვიყოთ, რადგან ისინი მაშინვე გადადის ბირთვის რეჟიმში და იღებს ოპერაციული სისტემის ყველა მონაცემთან სრულად შეღწევ-

ის უნარს. ამიტომ ამ რეჟიმში მომუშავე კომპონენტები კარგად უნდა კონტროლდებოდეს, ტესტირებოდეს და თავისუფალი იყოს შეცდომებისაგან.

ბირთვთან ჩვენ შეგვიძლია ვიმუშაოთ არა უშუალოდ, არამედ მხოლოდ მომხმარებლის რეჟიმის მეშვეობით, ამიტომ მოკლედ განვიხილოთ ეს უკანასკნელი.

▌ ***<u>სამომხმარებლის რეჟიმს</u>*** ბირთვის რეჟიმთან შედარებით გაცილებით ნაკლები პრივილეგიები აქვს. იგი შედგება ქვესისტემებისაგან, რომლებიც შეტანისა და გამოტანის მოთხოვნებს ბირთვის რეჟიმის დრაივერს სპეციალური ***შეტანა-გამოტანის მენეჯერის*** საშუალებით გადასცემს. მომხმარებლის დონე შედგება ***გარემოცვის*** (***Environment***) ***ქვესისტემებისა*** და ***ინტეგრირებული*** (***Integral***) ***ქვესისტემებისაგან.***

გარემოცვის ქვესისტემები საჭიროა სხვადასხვა სახის ოპერაციული სისტემისათვის დაწერილი გამოყენებების (პროგრამების) ასამუშავებლად. არც ერთ მათგანს არ შეუძლია პირდაპირ შეაღწიოს კომპიუტერის აპარატურულ ნაწილში, ხოლო მეხსიერების რესურსებში მათ აქვს შეზღუდული შეღწევის საშუალება. ამას ისინი ახდენს ***ვირტუალური მეხსიერების მენეჯერის*** მეშვეობით, რომელიც აქვს თითოეულ ოპერაციულ სისტემას. აღნიშნული მენეჯერი მუშაობს ბირთვის რეჟიმში და ამიტომ მას აქვს მეხსიერებაში შეღწევისათვის აუცილებელი პრივილეგია.

გარემოცვის ქვესისტემებია***Win32, POSIX, OS/2***, რომელთაგა-ნაც ყველაზე მნიშვნელოვანია ***Win32*** ქვესისტემა. იგი კომპიუტე-რულ პროგრამებს აწვდის ***გამოყენებათა დაპროგრამების ინტერფეისს*** (***<u>A</u>****pplication* ***<u>P</u>****rogramming* ***<u>I</u>****nterface*, ***API***). დარჩენილი ორი ქვესისტემაც აწვდის კომპიუტერულ პროგრამებს ***გამოყენებათა და პროგრამების საკუთარ ინტერფეისებს,*** მაგრამ სამომხმარებლო მოთხოვნების მისაღებად და შედეგების გასაცემად ისინი ***Win32***-ს იყენებს**.** მაშასადამე, ***Win32***-ს მუდმივად ფუნქციონირების გარეშე ***Windows*** ვერ იმუშავებს. ამით განსხვავდება იგი დანარჩენი ორი ქვესისტემისაგან. ***Win32*** ამუშავებს ყველაფერს, რომელიც დაკავშირებულია კლავიატურასთან, მაუსთან და მონიტორთან. ინტერაქტიული მომხმარებლის არარსებობის დროს იგი სერვერებზედაცაა საჭირო. ***Win32*** მართავს ფანჯრებსა და ბაზისურ სერვისებსაც.

გარემოცვის ***OS/2*** ქვესისტემა მხარს უჭერს ოპერაციული ***OS/2***სისტემის ***16***-თანრიგიან გამოყენებებს (პროგრამებს) და ახდენს ***OS/2 2.1.x*** სისტემის ემულირებას. გარემოცვის ***POSIX*** ქვესისტემა მხარს უჭერს ***POSIX.1*** სტანდარტის შესაბამისად დაწერილ გამოყენებებს (პროგრამებს).

გარემოცვის ინტეგრირებული ქვესისტემები თვალყურს ადევნებს ოპერაციული სისტემის რამდენიმე ფუნქციის შესრულებას. მათ შემადგენლობაში შედის „უსაფრთხოების სამსახურის ქვესისტემები“, „მუშა სადგურის ქვესისტემები» და „სერვისის სამსახურის ქვესისტემები“.

„უსაფრთხოების სამსახურის ქვესისტემები“მიმართავს შეღწევის მარკერებს, შესაძლებელს ხდის ან კრძალავს მომხმარებლის სააღრიცხვო ჩანაწერებთან მიმართვას, ამუშავებს ავტორიზაციის მოთხოვნებს და წარმოშობს სისტემაში მომხმარებლის შესვლის პროცესს. ***„მუშა სადგურის ქვესისტემები“*** კომპიუტერს უზრუნველყოფს ქსელში კომპიუტერის შეღწევას, ხოლო ***„სერვისის სამსახურის ქვესისტემები“***კომპიუტერს აწვდის ქსელურ სერვისებს.

■ ***<u>ბირთვისა და სამომხმარებლის რეჟიმებს შორის ძთავარი განსხვავებები.</u>*** სამომხმარებლო რეჟიმში მომუშავე პროგრამები ბირთვის რეჟიმში მომუშავე პროცესებისაგან იმით განსხვავდება, რომ მათ:

- არ შეუძლია მოწყობილობებში პირდაპირი შეღწევა;
- აპარატურული რესურსებს გამოყენებაზე მათ მიერ ფორმირებული ყველა მოთხოვნის შესრულების ნებართვა უნდა გასცეს ბირთვი რეჟიმის კომპონენტებმა;
- გამოეყოფა შეზღუდული ზომის სამისამართო სივრცე;
- ისინი შეიძლება ფიზიკური მეხსიერებიდან გადაიტვირთოს ხისტ დისკზე არსებულ ვირტუალურ მეხსიერებაში;
- მათ აქვს ბირთვის რეჟიმის პროცესებზე ნაკლები პრიორიტეტები, რაც ოპერაციულ სისტემას იცავს მწარმოებლურობის შეზღუდვისაგან ან გამოყენებების (პროგრამების) მიზეზებით წარმოშობილი შეყოვნებებისაგან.

■ ***<u>ოპერაციული სისტემის მუშაობის მაგალითი.</u>*** დავუშვათ, რომ გვინდა შევადგინოთ და დავბეჭდოთ რაიმე ტექსტი.

ამისათვის უნდა გავხსნათ ტექსტური რედაქტორი, ავკრიფოთ ტექსტი, შევინახოთ ეს დოკუმენტი და შემდეგ იგი დავბეჭდოთ პრინტერის საშუალებით. განვიხილოთ, ამ მანიპულაციების დროს თუ როგორ მოქმედებს ოპერაციული სისტემა.

უპირველეს ყოვლისა, კლავიატურა და მაუსი ჩვენ მიერ აკრეფილ მონაცემებს გადააგზავნის ოპერაციულ სისტემაში; ოპერაციული სისტემა გადაწყვეტს, რომ ტექსტური რედაქტორი წარმოადგენს აქტიურ პროგრამას და იყენებს მასთან მუშაობისათვის საჭირო მოდულებს. იგი მიიღებს ჩვენ მიერ ჩვენს მიერ შეტანილ მონაცემებს და მათ გადააგზავნის თავისთვის შესანახად. საქმე ისაა, რომ ოპერაციული სისტემა მცირე მოცულობისაა და ინფორმაციას მხოლოდ საკუთარი მუშაობისათვის ინახავს დროებით. კომპიუტერს უეცრად თუ გამოვრთავთ, მაშინ ოპერაციულ სისტემაში შენახული მთელი ინფორმაცია დაიკარგება. ამიტომ ინფორმაცია დისკზე უნდა იქნეს შენახული. თითოეული ჩვენი რედაქტირება ტექსტური რედაქტორიდან ოპერაციული სისტემით გაიგზავნება ცენტრალურ პროცესორში. იქ ბრძანებები გადაითარგმნება სამანქანო (ორობით) ენაზე. ***პროცესორი უკვე ბირთვის რეჟიმში მუშაობს.*** ამ დროის განმავლობაში ოპერაციული სისტემა მთელ ინფორმაციას ***გრაფიკული ბარათის*** დახმარებით ჩვენი კომპიუტერის მონიტორზე გვაწვდიდა. ამის შემდეგ მისი შენახვაა საჭირო, რისთვისაც თითი უნდა დავაჭიროთ შენახვის *Save (Сохранить)* ღილაკს. შენახვის მოთხოვნას ტექსტური რედაქტორი გაუგზავნის ოპერაციულ სისტემას. ეს უკანასკნელი ამ მოთხოვნის საპასუხოდ გვთავაზობს ამოვირჩიოთ ფაილის შენახვის ადგილი და ამ ფაილის სახელი. ჩვენგან საჭირო პარამეტრების მიღების შემდეგ ოპერაციული სისტემა ტექსტს შესანახად აგზავნის მუდმივი მეხსიერების მოწყობილობაში.

დგება ჩვენ მიერ შედგენილი ტექსტის დაბეჭდვის დრო. თითი უნდა დავაჭიროთ დაბეჭდვის *Print (Печать)*, რის შემდეგ ოპერაციული სისტემა მიიღებს ჩვენს ტექსტს და მას გადაუგზავნის ჩვენსავე მიერ მითითებულ სლოტზე მიერთებულ პრინტერს.

2. Windows-ში რეალიზებული ზოგიერთი ტექნოლოგია

Windows-ში პირველად იქნა გამოყენებული ***WYSIWYG*** (***W****hat* ***Y****ou* ***S****ee* ***I****s* ***W****hat* ***Y****ou* ***G****et* - *რასაც ხედავთ იმას მიიღებთ*) პრინციპი, რომელიც საშუალებას იძლე-

ვა მონიტორზე არსებულ გამოსახულებას სრულად შეესაბამებოდეს შემდგომში ქაღალდზე გადატანილი გამოსახულება.

ერთი დოკუმენტის მომზადების დროს ***Windows*** უზრუნველყოფს რამდენიმე პროგრამის ერთობლივ მუშაობას და გამოყენებებს შორის ობიექტების გადატანისა და კოპირების გზით საშუალებას გვაძლევს შევქმნათ განსხვავებული ტიპის მონაცემების შემცველი კომპლექსური დოკუმენტები. ამისათვის სისტემას აქვს ინტეგრაციის სპეციალური საშუალებები.

ყველაზე ხშირად გამოიყენება მეხსიერების ***გაცვლის ბუფერად*** (*clipboard*) წოდებული სპეციალური უბანი, რომელიც საჭიროა პროგრამებსა და დოკუმენტებს შორის მონაცემების გადასატანად. შეგვიძლია მოვნიშნოთ რომელიმე ობიექტი, შესანახად მოვათავსოთ იგი გაცვლის ბუფერში და შემდეგ იგი ჩავსვათ იმავე დოკუმენტის სხვა ადგილზე, ან როგორც იმავე, მეორე პროგრამის მიერ შექმნილ სხვა დოკუმენტში.

Windows-ში გამოყენებულია ***OLE*** (***O****bject* ***L****inkingand* ***E****mbedding - ობიექტების დაკავშირებისა და ჩანერგვის*) ტექნოლოგია, რომელიც საშუალებას გვაძლევს, სამუშაოს ნაწილი რედაქტირების ერთი პროგრამიდან გადავცეთ მეორე პროგრამას და შედეგები დავაბრუნოთ უკან. მაგალითად, კომპიუტერზე დაყენებულ ***სავამომცემლო სისტემას OLE*** ტექნოლოგიით შეუძლია გარკვეული ტექსტი დასამუშავებლად გაუგზავნოს ტექსტურ რედაქტორს, ან რომელიმე გამოსახულება – გამოსასახულების რედაქტორს. ***OLE***-ს გამოყენების ძირითადი უპირატესობა (გარდა ფაილის ზომის შემცირებისა) ის არის, რომ იგი საშუალებას გვაძლევს შევქმნათ მთავარი ფაილი, ფუნქციების კარტოთეკა, რომელსაც შეუძლია მიმართოს პროგრამამ ამ ფაილს შეუძლია მოახდინოს ოპერირება საწყისი პროგრამიდან მიღებულ მონაცემებზე, დაამუშაოს ისინი და შემდეგ დამუშავებული სახით დაუბრუნოს საწყის პროგრამას.

***OLE*-ტექნოლოგია** საშუალებას გვაძლევს ერთ დოკუმენტში გავაერთიანოთ აბსოლუტურად განსხვავებული *insertion*წარმოშობის ობიექტები, მაგალითად, ტექსტი, ფოტოსურათი და მუსიკა. ასეთი ობიექტები შეგვიძლია ჩავნერგოთ ან გაცვლის ბუფერის გავლით ან კონტექსტური მენიუს ბრძანებით: *Paste* ►*Object* [*Вставка* ► *Объект*].

OLE ტექნოლოგიას ***1996*** წელს ეწოდა ***ActiveX***ტექნოლოგია. ***ActiveX*** არის დაპროგრამების სხვადასხვა ენაზე დაწერილი პროგრამებიდან გამოსაყენებლად ვარგისი პროგრამული კომპონენტების განმსაზღვრელი ფრეიმვორკი. ***ფრეიმვორკი*** (*framework – მზიდი კონსტრუქცია, კარკასი, სტრუქტურა*) პროგრამაა, რომელიც აიოლებს დიდი პროგრამული პროექტის დამუშავებისა და გაერთიანების პროცესს.

გამოყენებებს შორის ურთიერთზემოქმედებისათვის ***Windows***-ში არსებობს მონაცემების დინამიკურად გაცვლის ***DDE*** (***D****ynamoc* ***D****ata* ***E****xhange*) პროგრამული პროტოკოლი. იგი მონაცემების შემკრებ პროგრამას საშუალებას აძლევს, მონაცემები სხვა გამოყენებებს რეალურ დროში სინქრონულად გაუნაწილოს.

ოპერაციულ სისტემა ***Windows***-ი ***ფაილების კონვერტირების*** ანუ დოკუმენტების ფორმატის საშუალებასაც გვაძლევს. ამისათვის მრავალი გამოყენება გარკვეული პროტოკოლების მიხედვით მონაცემები ერთი დოკუმენტიდან მეორეში გადაგზავნისათვის შეიცავს იმპორტ-ექსპორტის ***სპეციალურ ფილტრებს***. მათი საშუალებით, მაგალითად, ტექსტური ფაილი გარდაიქმნება ***Word***-ის დოკუმენტად და პირიქით.

3. სპეციფიკური ცნობები ინტერფეისის შესახებ

ინტერფეისი *საშუალებების ერთობლიობაა, რომლის დახმარებითაც მომხმარებელი ურთიერთობს კომპიუტერთან, ხოლო ამ უკანასკნელის აპარატურული და პროგრამული უზრუნველყოფები – ერთმანეთთან (ინგ. «**Inteface**» - ქართ. „ურთიერთქმედება“, „შეერთება“, „შინავანი ზედაპირი“). განასხვავებენ აპარატურულ, პროგრამულ და სამომხმარებლო (მომხმარებლის) ინტერფეისებს. მოკლედ განვსაზღვროთ თითოეული მათგანი.*

■ ***აპარატურული ინტერფეისი*** *წარმოადგენს სალტეების, გასართების, მათანხმებელი მოწყობილობების, ალგორითმებისა და პროტოკოლების სისტემას, რომელიც უზრუნველყოფა კავშირს კომპიუტერული სისტემის ნაწილებს შორის. აპარატურულ ინტერფეისის მახასიათებლებზეა დამოკიდებული სისტემის სწრაფმოქმედება და საიმედოობა.*

■ ***პროგრამული ინტერფეისი*** *(Programming Interface) დამპროგრამებელ მოწყობილობებთან ან გამოყენებით პროგრამებთან მომხმარებლის ურთიერთობისა და თავად პროგრამებს შორის ინფორმაციის გაცვლის ხერხია. პროგრამების ოპტიმალური პარამეტრების რეალიზების გზით იგი განსაზღვრავს აღნიშნული ურთიერთობის ფუნქციურობასა და მოხერხებულობას. პროგრამული ინტერფეისის სპეციფიკური სახეა* ***გამოყენებათა პროგრამული ინტერფეისი*** *ანუ* ***API*** *(იხ. გვ. **101**). მისი საშუალებით გარე პროგრა-*

*მული უზრუნველყოფები ოპერაციული სისტემისაგან იღებს მზა კლასების, პროცედურების, ფუნქციების, სტრუქტურებისა და კონსტანტების ნაკრებს. სატოვნად **API** შეიძლება შევადაროთ სპეციფიკურ პროგრამულ„შტეფსელს»", რომელშიც „ჩართული" გარე პროგრამული პროდუქტები ზემოთ აღნიშნული ნაკრებს იღებს ოპერაციული სისტემისაგან.*

■ ***სამომხმარებლო (მომხმარებლის) ინტერფეისი** (**U**ser **I**nterface - **UI**) წარმოადგენს პროგრამული და აპარატურული საშუალებების ერთობლიობას, რომელიც უზრუნველყოფს კომპიუტერთან მომხმარებლის ურთიერთობას. ეს ურთიერთობა ეწუძნება **დიალოგს**. მოცემულ შემთხვევაში **დიალოგად** მიიჩნევა ადამიანსა და კომპიუტერს შორის ინფორმაციის რეგლამენტირებული გაცვლა, რომელიც დროის რეალურ რეჟიმში ხდება და მიმართულია კონკრეტული ამოცანების ერთობლივი გადაწყვეტისაკენ. თითოეული დიალოგი შედგება შეტანა/გამოტანის ცალკეული პროცესებისაგან, რომლებიც ერთმანეთთან ფიზიკურად აკავშირებს მომხმარებელსა და კომპიუტერს. ინფორმაცია შეტყობინებების გაცვლის გზით გაიცვლება. არსებობს შემდეგი სახის **სამომხმარებლო ინტერფეისი**:*

*-**საკომანდო ინტერფეისი.** ინტერფეისის ამ სახის დროს ადამიანი კომპიუტერს გადასცემს „ბრძანებებს", ხოლო კომპიუტერი ასრულებს მათ და შედეგებს აწვდის ადამიანს. საკომანდო ინტერფეისი რეალიზებულია პაკეტური ტექნოლოგიისა და **საკომანდო სტრიქონის** ტექნოლოგიების სახით;*

*- **WIMP-ინტერფეისი** (**W**indow - ფანჯარა, **I**mage- სახე, **M**enu – მენიუ, **P**ointer - მაჩვენებელი). მისი დამახასიათებელი თავისებურებაა ის, რომ მომხმარებელთან დიალოგი მიმდინარეობს არა ბრძანებების მეშვეობით, არამედ **გრაფიკული სახეების** – მენიუს, ფანჯრებისა და სხვა ელემენტების დახმარებით. ამიტომWIMP-ინტერფეისს ხშირად **გრაფიკულ ინტერფეისსაც** უწოდებენ. ბრძანებები კომპიუტერს ამ ინტერფეისის დროსაც გადაეცემა, მაგრამ ისინი გრაფიკული სახეებითაა წარმოდგენილი. ასეთი სახეების გამოყენება უფრო მოსახერხებელია მომხმარებლისათვის, რადგან მას არ სჭირდება მრავალი ბრძანების შესწავლა.*

*- **SILK-ინტერფეისი** (**S**peech - საუბარი, **I**mage – სახე, **L**anguage - ენა, **K**nowlege– ცოდნა). ამ სახის ინტერფეისი ყველაზე უფროა მიახლოებული ურთიერთობის ადამიანურ ფორმასთან. მის ჩარჩოებში ადამიანი და კომპიუტერი ჩვეულებრივ ერთმანეთს „ესაუბრებიან", რომლის დროსაც კომპიუტერი აანალიზებს ადამიანურ ლაპარაკს, მასში პოულობს საკვანძო ფრაზებს, რომლებსაც მისთვის გადაცემულ ბრძანებებად აღიქვამს და ასრულებს მათ. კომპიუტერი შესრულების შედეგებსაც აძლევს ადამიანისათვის გასაგებ ფორმას. ეს ინტერფეისი მკაცრ მოთხოვნებს უყენებს კომპიუტერის აპარატუ-*

რულ რესურსებს და ამიტომ იგი ჯერჯერობით ძირითადად სამხედრო მიზნებისთვისაა გამოყენებული.

ზემოთ აღნიშნულიდან გამომდინარე, თანამედროვე კომპიუტერებში მასობრივად გამოიყენებულ ინტერფეისებს წრმოადგენს ***საკომანდო სტრიქონი*** *და* ***გრაფიკული ინტერფეისი****. მასობრივი მომხმარებლისათვის გამოსაყენებლად უფრო მოსახერხებელია გრაფიკული ინტერფეისი. ინტერფეისების სფეროს ცნობილ სპეციალისტს,* ***2012*** *წელს გამოსული წიგნის „****The Linux Command Line: A Complete Introduktion****“ (აღნიშნული წიგნი სახელწოდებით „****Командная строка Linux. Полноеруководство****“* ***2017*** *წელს რუსულ ენაზეც გამოვიდა) ავტორის* ***უილიამ შოტსის*** *(****William Shotts****) აზრით* ***გრაფიკული ინტერფეისი*** *მარტივი ამოცანების გადაწყვეტას კიდევ უფრო ამარტივებს, ხოლო* ***საკომანდო სტრიქონი*** *რთული ამოცანების გადაწყეტის საშუალებას გვაძლევს. აღნიშნულის გამო ორივე ეს ინტერფეისი მშვენივრად თანაცხოვრობს თანამედროვე პერსონალურ კომპიუტერებში. მაგალითად, გრაფიკულ ინტერფეისიან ოპერაციულ სისტემად მიჩნეულ* **Windows** *სისტემებში თავისუფლად შეგვიძლია* ***საკომანდო სტრიქონიც*** *ავამუშაოთ.* **Windows 7***-ის შემთხვევაში ამისათვის უნდა დავაწვეთ ღილაკების Win+R შეხამებას, მენიუდან ამოვირჩიოთ ოფცია* **cmd (*Командная строка*)** *და, ბოლოს, ვიმოქმედოთ* **OK** *ღილაკზე.* ***ანალოგურად, ტრადიციულად საკომანდო სტრიქონიანი ოპერაციული Linux სისტემა გრაფიკული ინტერფეისითაც არის აღჭურვილი.***

მასობრივი მომხმარებლის ამოცანათა უმრავლესობა მარტივ ამოცანათა კლასს მიეკუთვნება, რომელთა გადასაწყვეტად გრაფიკული ინტერფეისია უბადლო, ამიტომ ქვემოთ მხოლოდ მასზე გავამახვილებთ ყურადღებას.

4. Windows-ის გრაფიკული ინტერფეისი

ოპერაციული სისტემა **Windows**მოიცავს გრაფიკული იტერფეისის ექვს ძირითად ელემენტს – სამუშაო მაგიდას, ამოცანების პანელს, ფანჯრებს, მენიუს ინსტრუმენტების პანელსა და პიქტოგრამებს. გავეცნოთ მათ.

1. <u>სამუშაო მაგიდა</u> **(*Desktop*)**. კომპიუტერის ჩართვის შემდეგ დისკიდან ოპერატიულ მეხსიერბაში გადმოვა ოპერაციული სისტემის ბირთვი, რომელიც ეკრანზე გაშლის სამუშაო მაგიდას. იგი ოპერაციული სისტემის თავისებური სატიტულო ფურცელია, რომელზეც შეიძლება განთავსდეს ინტერფეისის დანარჩენი ელემენტები: სისტემის მთავარი მენიუს მქონე ამოცანათა პანელი, სხვადასხვა ფანჯარა, ინსტრუმენტების პანელები და სხვადასხვა ნიშანი.

2. <u>ამოცანების პანელი</u> *(taskbar)* ინტერფეისის უმნიშვნელოვანესი ელემენტია, რომელიც გვიჩვენებს მოცემულ მომენტში რეალიზე-

ბად პროგრამებს და აადვილებს მათ შორის შესასრულებელ გადართვებს. ზოგიერთი სისტემური პროგრამის გარდა პრაქტიკულად ნებისმიერი პროგრამა ჩატვირთვისას ამოცანების პანელზე ჩნდება ვირტუალური ღილაკის სახით, რომელზეც მაუსის მარცხენა ღილაკის ზემოქმედება ააქტიურებს პროგრამას. პროგრამის მუშაობის დასრულებისას მისი ღილაკი ქრება ამოცანათა პანელიდან.

ჩვეულებრივ ***ამოცანათა პანელი*** რუხი ფერისაა და ოპერაციული სისტემის სტანდარტული აწყობისას იგი სამუშაო მაგიდის ქვედა ნაწილზეა განთავსებული. ადგილმდებარეობის შესაცვლელად საჭიროა ამოცანების პანელის ანაწყობში გამოვრთოთ ოფცია (მენიუს ელემენტი) ***„Lock the taskbar“*** (*„закрепить панель задач“*), პანელის თავისუფალ ადგილზე მივიტანოთ მაუსის კურსორი, თითი დავაჭიროთ მაუსის მარცხენა ღილაკს და მისგან თითის აუშვებლად მაუსით გადავათრიოთ პანელი მუშა მაგიდის ჩვენთვის სასურველ ადგილზე.

მოცემულ მომენტში გააქტივებული პროგრამების გასაკონტროლებლად და მათ შორის გადართვების გასაადვილებლად ***ამოცანების პანელი*** ეკრანზე პრაქტიკულად ყოველთვის გამოჩენილია, მაგრამ საჭიროებისამებრ მისი დამალვაც შეგვიძლია. ამისათვის უნდა ჩავრთოთ ოფცია *„Automatically hide the taskbar“* (*„Автоматрчески скрывать панель задач“*). ეკრანზე მისი ასახვის საჭიროებისას მაუსის კურსორი უნდა მივიტანოთ სამუშაო მაგიდის იმ მხარეზე, სადაც იგი იქნა დამალული.

3. <u>ფანჯარა</u> (*Window*).ამოცანების პანელის გარდა სამუშაო მაგიდაზე შეიძლება იყოს ერთი ან რამდენიმე ფანჯარა, რომლებიც შეგვიძლია მოზაიკურად ან კასკადურად განვათავსოთ. ***მოზაიკურად განთავსებისას*** ფანჯრები გარკვეული თანამიმდევრობითაა დალაგებული ეკრანზე, ხოლო ***კასკადურად განლაგებისას*** ისინი ერთმანეთზეა „დაყრილი“. ფანჯარა ეკრანის სპეციალურად გაფორმებულ და მოჩარჩოებულ არეა, რომელზეც განთავსებულია ზემოქმედებებისათვის განკუთვნილი სხვადასხვა ობიექტი. ჩვენ შეგვიძლია ფანჯარა გავხსნათ, დავხუროთ, ჩავკეცოთ, გავშალოთ და გადავადგილოთ. ოპერაციული ***Windows*** სისტემაში არსებობს შემდეგი სამი ტიპის ფანჯარა:

■ ***გამოყენებების (პროგრამების) ფანჯარა***. იგი შეიცავს ***3*** სტანდარტულ ელემენტს: პროგრამათა სათაურებსა და მენიუს, ინსტრუმენტების ერთ ან რამდენიმე პანელს და მდგომარეობის სტრიქონს. ***Windows*** საშუალებას გვაძლევს, ერთდროულად ავამოქმედოთ რამდენიმე გამოყენება (პროგრამა) და მონაცვლეობით ვიმუშაოთ მათთან: ერთი პროგრამიდან გადავერთოთ მეორეზე. აქტიური ფანჯარა ყოველთვის დანარჩენი ფანჯრებს „დაედება“ და მხოლოდ მას შეუძლია მომხმარებლის ბრძანებების აღქმა;

■ ***დოკუმენტების (პროგრამათა დამუშავების ობიექტების) ფანჯარა***, რომელიც აუცილებლად შეიცავს სათაურს;

■ ***სადიალოგო (დამუშავებისათვის საჭირო ინსტრუმენტების) ფანჯარა***, რომელთა სათაურები ჩვეულებრივ ემთხვევა მათი გამღებ ბრძანებათა სახელწოდებებს.

ზოგიერთი ფანჯარა შეიცავს ინტერფეისის დამატებით ელემენტებს: სახაზავებს, გადხვევის (დატრიალების) ზოლებს, მდგომარეობის სტრიქონებს, საკომანდო ღილაკებს ან სიებს.

Windows-ის ბევრი გამოყენება მრავალფანჯრულია, შეუძლია შეიცავდეს რამდენიმე ჩალაგებულ ფანჯარას. ასეთია, მაგალითად, ბრაუზერი ***Internet Explorer***.

გამოყენებებისა და დოკუმენტების ფანჯრები შეგვიძლია წარმოვადგინოთ: ***სრულეკრანულად*** (ფანჯარა დაიკავებს მთელ ეკრანს), ***ჩვეულებრივად*** (ფანჯარა დაიკავებს ეკრანის ნაწილს) და ***ჩაკეცილად*** (ფანჯარა პანელზე არსებულ ღილაკშია „ჩაკეცილი“).

ფანჯრის ზედა ნაწილი შეიცავს სათაურს, რომელშიც ჩვეულებრივ მოთავსებულია პროგრამის და მასში გახსნილი დოკუმენტის სა-ხელი. სათაურის არეში მაუსის მარცხენა ღილაკის დაჭერის შემდეგ იგი შეიძლება გადავანაცვლოთ სამუშაო მაგიდის ფარგლებში.

ფანჯრის მარჯვენა ზედა კუთხეში არის მართვის ***3*** ღილაკი. მარცხენა ღილაკი გამოიყენება ფანჯრის ჩასაკეცად (***ჩაკეცოს, Свернуть, Zoom out***)ამოცანების პანელში;შუა ღილაკი - ფანჯარას ეკრანის მთელ ზომაზე გასაშლელად და საწყისი ზომის აღსადგენად. ფანჯრის გარეგანი სახე მის მდგომარეობაზე დამოკიდებულებით იცვლება.

სადიალოგო ფანჯარას ჩვეულებრივ ერთადერთი მმართველი ღილაკი (***დაიხუროს, Close, Закрыть***) აქვს.

ფანჯრის ჩარჩო ფანჯრის ზომების შეცვლის საშუალებას გვაძლევს.

4. <u>მენიუ</u> (***menu***) ჩვეულებრივ ფანჯრის სათაურის ქვემოთაა მოთავსებული; იგი ინტერფეისის მნიშვნელოვანი ელემენტია, რომელსაც აქვს რამდენიმე დონე და შეიძლება შეიცავდეს მოცემულ მომენტში ამოსარჩევად მიუწვდომელ ობიექტებს. ოპერაციულ სისტემა ***Windows***-ში გამოიყენება შემდეგი ***4*** სახის მენიუ:

■ ***სისტემის მთავარი მენიუ***, რომლის გასახსნელად მაუსის მარცხენა კლავიში უნდა დავაჭიროთ ამოცანების პანელის მარცხენა კუთხეში არსებულ ***Start (Пуск)*** ღილაკზე ან ვიმოქმედოთ კლავიატურის ***Windows***-ის ლოგოტიპიან ***WL*** კლავიშზე (*მოთავსებულია* ***ALT****და* ***Ctrl*** *კლავიშებს შორის*). სტანდარტულად აწყობილი ოპერაციული სისტემის დროს იგი შეიცავს შვიდ პუნქტს: *All Programs, Documents, Control Panel, Search, Help and Support, Run, Shut down* (*Программы, Документы, Настройка, Пойск, Справка, Выполнить, Завершение работы*).შესაბამისად, მთავარი მენიუ საშუალებას გვაძლევს ავამუშაოთ პროგრამა, გავხსნათ დოკუმენტი, შევცვალოთ სისტემის აწყობა, ვიპოვოთ საჭირო ობიექტი, მივიღოთ ცნობები და დავამთავროთ ***Windows***-ის მუშაობა.

■ ***პროგრამების მენიუ***რომელიც განთავსებულია თითოეულ ამუშავებულ პროგრამაში (გამოყენებაში). იგი ჩვეულებრივად იკავებს პროგრამის ფანჯრის სახელწოდების ქვეშ არსებულ მეორე სტრიქონს. პროგრამების მენიუს ზოგიერთ პუნქტებსაც აქვს მათდამი დაქვემდებარებული მენიუ. ისინი მათი ამორჩევის დროს იხსნება. ***დოკუმენტებთან მომუშავე თითოეულ გამოყენებაში*** არსებობს მენიუ *File* (*Файл*), ხოლო ***მონაცემებთან მომუშავე ბევრ გამოყენებაში*** – მენიუ *Edit*(*Правка, Редактирование*). საცნობარო სისტემაში შეიძლება შევაღწიოთ მენიუთი*Help* (*Справка*). მენიუს მრავალი ბრძანება შეიძლება გამოვიძახოთ კლავიატურის კლავიშებით, მაგალითად, ნებისმიერი პროგრამა შეიძლება დავასრულოთ კლავიშების *Alt + F4*კომბინაციით.

■ ***ობიექტების კონტექსტური მენიუ*** ***Windows***-ის პრაქტიკულად ყველა გამოყენებაში მაუსის მარჯვენა კლავიშზე დაჭერით გაიხსნება და აქტიური ობიექტისათვის გამოსაყენებელ ბრძანებებს შეიცავს.

■***გამოყენებებისა და დოკუმენტების მმართველი მენიუ*** (***სისტემური მენიუ***) გაიხსნება ფანჯრის ზედა მარცხენა კუხეში არსებულ ნიშანზე მაუსის მარცხენა კლავიშის დაჭერით ან კლავიატურის

კლავიშების *Alt + Пробел* კომბინაციით. ეს მენიუ მართავს ფანჯრებს და ახდენს ფანჯრების მმართველი (*ჩაკეცვა, გაშლა/აღდგენა, დახურვა*) ღილაკების დუბლირებას. სისტემური მენიუს ნიშანზე მაუსის ორმაგი დაწკაპუნება დახურავს აქტიურ ფანჯარას.

5. <u>ინსტრუმენტების პანელი</u> (*Toolbar*). გამოყენებების ფანჯარა შეიძლებაშეიცავდეს ერთ ან რამდენიმე ***„ინსტრუმენტების პანელს“***; თითოეულს მათგანს აქვს მოცემული პროგრამის გარკვეული ფუნქციების შესასრულებლად საჭირო ღილაკების ნაკრები. მაგალითად, *Exsplorier* (*Проводник*) პროგრამის ინსტრუმენტების *Normal* (***Обычный***) პანელს აქვსსაქაღალდეებს შორის გადასვლის, ობიექტების კოპირების, გადანაცვლების, ძიებისა და გაძევების სტანდარტული ოპერაციების შესასრულებლად საჭირო ღილაკები. გამოყენებით პროგრამებში ინსტრუმენტების პანელი *სტანდარტული* (*Toolbar-Seandart*)განკუთვნილია ზოგადი სახის (დოკუმენტის შექმნის, გახსნის, შენახვის, ბეჭდვისა და ა.შ. ბრძანებების შესასრულებლად.

6. <u>პიქტოგრამები</u>.საგნის, ობიექტის ან მოვლენის სქემატურ გამოსახულებებში მათი ცალსახად ამოცნობისათვის საჭირო თვისებები აისახება, რის გამოც აღნიშნულ გამოსახულებებს „დახატულ ჩანაწერებსაც“უწოდებენ. სიტყვათა შეთანხმების ***„დახატული ჩანაწერი“*** ერთი სიტყვით გამოსახატავად ლათინური და ბერძნული სიტყვების ურთიერთმიწერით სპეციალურად შეიქმნა ლამაზი ხელოვნური სიტყვა ***პიქტოგრამა*** (ლათ. *pictus* – დახატული და ბერძ. γράμμα- ჩანაწერი). კომპიუტერის გრაფიკული ინტერფეისის ნიშნები პიქტოგრამების ერთ-ერთი ნაირსახეობაა.

გრაფიკულ ოპერაციულ სისტემაში ყველა ობიექტს აქვს სტანდარტული (ჩვეულებრივ ***32×32*** პიქსელის) ზომის კვადრატული პიქტოგრამა (სურათი). პიქტოგრამის მიხედვით ხშირად შეგვიძლია განვსაზღვროთ ობიექტის ტიპი: იგი საქაღალდეა, პროგრამაა, დოკუმენტია, იარლიყია თუ სხვა რაიმე.

■ ***საქაღალდე*** (***MS-DOS***-ში არსებული კატალოგის ანალოგი), ნებისმიერი ელემენტის (სხვასაქაღალდეების, ფაილებისა და იარლიყების) შესანახი კონტეინერია. არსებობს სამომხმარებლო საქაღალდეები და ***სისტემური საქაღალდეები***, რომლებსაც თავად ოპერაციული სისტემა ქმნის და ემსახურება. ოპერაციული სისტემის

სტანდარტულად აწყობის დროს სამუშაო მაგიდაზე აუცილებლად არსებობს შემდეგი ***4*** სახის ***სისტემური საქაღლდე:***

1. ჩემი კომპიუტერი (***My computer***), რომელიც მოიცავს პერსონალური კომპიუტერის ყველა მოწყობილობას და უზრუნველყოფს კომპიუტრის ყველა რესურსთან შეღწევას (შესაბამისი უნივერსალური პროგრამა ***მეგზურის***, ანუ ***Exsplorier***–ის დახმარებით);

2. ჩემი დოკუმენტები (***My documents***), რომელშიც შეინახება კომპიუტერში არსებული ყველა ისეთი დოკუმენტი, რომელსაც მომხმარებელი არ ინახავს რომელიმე სხვა (ფარულ) ადგილზე.

3. ნაგვის კალათა (***Recycle Bin***), ესაა ხისტ დისკზე არსებული მეხსიერების შეზღუდული მოცულობის (სულ მცირე, 1%-ის ტოლი) მიდამო, რომელშიც შეინახება ყველა გაძევებული ობიექტი, რომლის ხელახლა აღდგენა შესაძლებელია.

4. ქსელური მიდამო (*The network environment*); მასში შენახულია ყველა ხელმისაწვდომი რესურსის (სერვერების, მუშა სადგურების, პრინტერებისა და ა.შ) პიქტოგრამები.

■ ***იარლიყი*** (*Link*) გარკვეული ობიექტის (საქაღალდის, პროგრამის, დოკუმენტის ან მოწყობილობის) მაძიებლის შემცველი სპეციალური ფაილია. თავად ობიექტი შეიძლება მომხმარებლისაგან შორს იყოს, ამიტომ იარლიყი უზრუნველყოფს მასში მოსახერხებელშეღწევას. იარლიყის არსებობა არ ცვლის თავად ობიექტის ადგილმდებარეობას, არამედ მხოლოდ მასში შეღწევას ამარტივებს.

5. Windows-ის პროგრამები (გამოყენებები)

ოპერაციულ სისტემა ***Windows***-ის სტრუქტურაში გაერთიანებულია უამრავი პროგრამა. განვიხილოთ მათი ამუშავებისა და დასრულების ხერხები.

არსებობს ***Windows-ის პროგრამების ამუშავების შემდეგი ხუთი ხერხი:***

1. სისტემის მთავარი მენიუს საშუალებით პროგრამის ამორჩევის ხერხი: *Start* ► *All programs* ► **…**; (*Пуск* ► *Программы* ► …);

2. სამუშაო მაგიდაზე არსებულ იარლიყზე დაწკაპუნების ხერხი;

3. „ცხელ“ კლავიშზე ან მათ კომბინაციაზე ზემოქმედების ხერხი (პროგრამისათვის თუ ამის საშუალებას მოცემული სისტემა იძლევა);

4. შესაბამის პროგრამაში დამუშავებადი დოკუმენტის გახსნის ხერხი;

5. ამოცანების პანელზე (*taskbar*) არსებულ ღილაკზე დაწკაპუნების ხერხი (*პროგრამა თუ სრულდებოდა და ჩაიკეცა*).

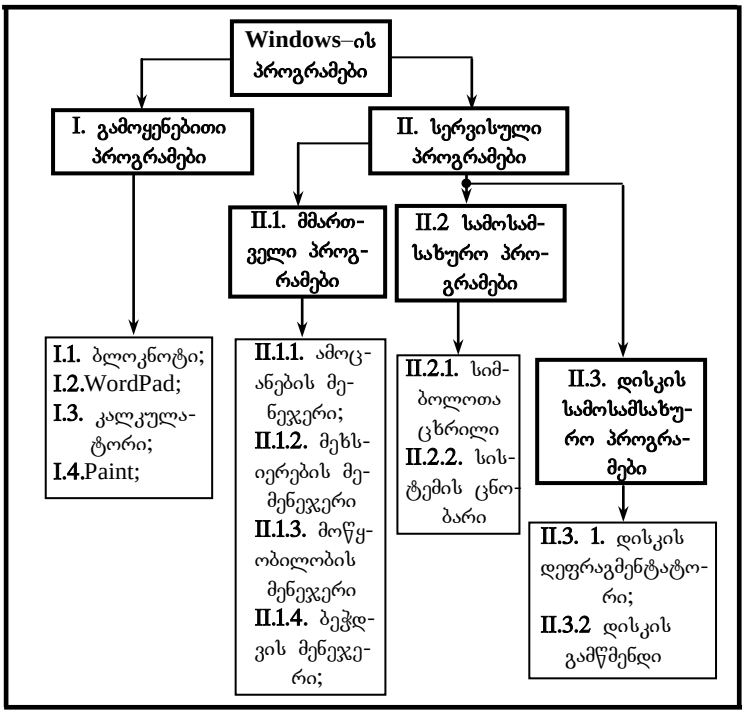
ასევე ხუთი ხერხით შეგვიძლია ***Windows-ის პროგრამების დახურვა:***

1. პროგრმის მენიუს შემდეგი ბრძანების შერსულების ხერხი:
File►Close ; (*Файл► Закрыть*);

2. პროგრამის ფანჯრის სათაურში არსებულ მმართველ *Close*(*Закрыть*) ღილაკზე დაჭერის ხერხი;

3. პროგრამის ფანჯრის სისტემურ მენიუში ბრძანება *Close*(*Закрыть*) ამორჩევის ხერხი (გამოიძახება *Alt + Пробел* კლავიშების კომბინაციით);

4. ნებისმიერი ობიექტის დახურვის კლავიშების სტანდარტულ *Alt +F4* კომბინაციაზე დაჭერის ხერხი;



ნახ.3.1. Windows-*ის სტრუქტურის პროგრამების კლასიფიკაცია*

5. *კონტექსტურ მენიუში Close(Закрыть) ამორჩევის ხერხი.*

Windows-ისთვის დამუშავებულია უამრავი პროგრამა. რომელთა კლასიფიკაცია ***3.1*** ნახაზზეა მოყვანილი. როგორც ამ ნახაზიდან ჩანს, აღნიშნული პროგრამები იყოფა გამოყენებით და სერვისულ პროგრამებად; განვიხილოთ ისინი.

I. გამოყენებითი პროგრამები (application program)გამოყენების კონკრეტულ სფეროში მონაცემების დამამუშავებელი, პროგრამებია, რომლებსაც მომხმარებელი საკუთარი პრობლემების გადასაწყვეტად იყენებს. მათთან შეიძლება მთავარი მენიუდან შევაღწიოთ შემდეგი ბრძანებით: *Start ► All programs ► Standard,*

(*Старт ►Программы ►Стандартные*).

ოპერაციულ სისტემა ***Windows***-ის სხვადასხვა ვერსიაში სხვადასხვა გამოყენებითი პროგრამა არსებობს, ოღონდ ყოველ მათგანში ყოველთვის გვხვდება შემდეგი ოთხი მათგანი:

1.1. ბლოკნოტი (***Notepad***). ბლოკნოტი ტექსტური რედაქტორია, რომელიც გამოიყენება ტექსტური ფაილების გადათვალიერებისათვის. იგიამუშავდება მენიუს ბრძანებით

Start ►All programs ► Standard ►Notepad;

(*Старт►Программы►Стандартные ► Блокнот*).

შექმნილი დოკუმენტის დისკზე შენახვისათვის საჭიროა მივუთითოთფაილის ახალი სათაური (საწინააღმდეგო შემთხვევაში იგი მას შეინახავს სახელით: *unnamed.txt*ანუ *უსათაურო. txt*).ბლოკნოტი იშვიათად გამოიყენება ტექსტური დოკუმენტების შესაქმნელად, რადგან იგი ტექსტის დაფორმატების საშუალებას არ გვაძლევს. გარდა ნაკლისა, ეს, მისი უდავო უპირატესობაცაა, რადგან საშუალებას გვაძლევს, ავიღოთ ნებისმიერი დაფორმატებული ტექსტი (მაგალითად *Word*-ის დოკუმენტი ან ინტერნეტის გვერდი), გაცვლის ბუფერით მოვათავსოთ ბლოკნოტში და შემდეგ უკვე დაუფორმატებლად (სუფთა ტექსტი) ჩავსვათ ნებისმიერ სხვა დოკუმენტში.

I.2. WordPad ტექსტური რედაქტორია, რომელიც ტექსტური დოკუმენტების არა მარტო რედაქტირების, არამედ დაფორმატების საშუალებასაც გვაძლევს. ამიტომ ბლოკნოტისაგან განსხვავებით *WordPad*-ის ფანჯარა შეიცავს ინსტრუმენტების პანელს, რომელზეც არსებობს დაფორმატებისათვის საჭირო ინსტრუმენტები. პროგრამა ბლოკნოტი ამუშავდება მენიუს ბრძანებით:

Start ▶ *All programs* ▶ *Standard* ▶ *WordPad*,

(*Старт* ▶ *Программы* ▶ *Стандартные* ▶ *WordPad*).

WordPad წარმოადგენს მძლავრი ***Microsoft Word*** პროცესორის გამარტივებულ ვერსიას.

I.3. კალკულატორი (*calculator*) რიცხვებზე გამოთვლების ჩასატარებლადაა განკუთვნილი (მისი ფუნქციები და საშუალებები არ განსხვავდება ჩვეულებრივი ელექტრონული კალკულატორის ფუნქციებისა და საშუალებებისაგან). პროგრამა ამუშავდება მენიუს ბრძანებით: *Star t*▶*Allprograms* ▶ *Standard* ▶ *Calculator*,

(*Старт* ▶ *Программы* ▶ *Стандартные* ▶ *Калькулятор*).

I.4. Paint ვექტორული გრაფიკით შესრულებული და რასტრული გრაფიკის ფორმატში შესანახი გამოსახულებების შექმნისა და რედაქტირებისათვის განკუთვნილი უმარტივესი გრაფიკული რედაქტორია. პროგრამაამუშავდებამენიუსბრძანებით:

Start ▶*All programs* ▶ *Standard* ▶ *Paint*,

(*Старт* ▶ *Программы* ▶ *Стандартные* ▶ *Paint*).

პროგრამის ფანჯარა შეიცავს მართვის ელემენტებს: მენიუს სტრიქონს, ინსტრუმენტების პანელს, ფერად პალიტრას. მისი საშუალებით შეგვიძლია ტექსტების სათაური შევქმნათ.

გამოყენებითი პროგრამების შემადგენლობაში არის სხვა (თამაშების, ელექტრონული ფოსტის, მულტიმედიისა და ა. შ.) პროგრამებიც.

II. სერვისული პროგრამები (***Services program***; ***Tool***) გამოყენებითი პროგრამებისაგან განსხვავებით განკუთვნილია პერსონალური კომპიუტერისა და თავად ოპერაციული სისტემის მომსახურებისათვის. ისინი საშუალებას გვაძლევს ვიპოვოთ და აღმოვფხვრათ ფაილური სისტემის დეფექტები, მოვახდინოთ აპარატურული და პროგრამული უზრუნველყოფის ოპტიმიზება და კომპიუტერის მომსახურებასთან დაკავშირებული ზოგიერთი რუტინული ოპერაციის ავტომატიზება. არსებობს ორი სახის - ***მმართველი*** და ***სამომსახურო*** პროგრამები, ოღონდ სამომსახურეო პროგრამებისაგან განცალკევებულად დგას ***დისკის სამომსახურო პროგრამები***. (იხ. ნახ. ***3.1***) განვიხილოთ თითოეული მათგანი.

ოპერაციულ სისტემა ***Windows***-ში გვაქვს შემდეგი ოთხი **მმართველი სერვისული პროგრამა**(იხ. ნახ. ***3.1***):

II.1.1. ამოცანების მენეჯერი (***Windows Task Manager***) ***Windows***-ის ყველა პროცესის მაკონტროლებელი ცენტრალური მმართველი პროგრამაა. ***Windows***-ის მუშაობის სეანსი იწყება ამოცანების მენეჯერის ამუშავებით, რომელიც ყველა სხვა პროგრამების ამუშავებისა და დასრულების ინიციატორია. ამოცანების მენეჯერი მართავს რამდენიმე ერთდროულად მომუშავე პროგრამას: მათ შორის ანაწილებს კომპიუტერის რესურსებს და აღნიშნულ პროგრამებს აძლევს ერთი ამოცანიდან მეორეზე გადართვის საშუალებას. ამოცანების მენეჯერის მუშაობის დასრულებით მთავრდება ***Windows***-ის მუშაობის სეანსიც;

II.1.2. მეხსიერების მენეჯერი (***The memory manager***) მართავს კომპიუტერის მეხსიერების განაწილების პროცესს;

II.1.3. მოწყობილობების მენეჯერი (***Device Manager***) მართვის კონსოლის აღჭურვილობა ანუ აპლეტია(ინგ.*applet* მიღებულია სიტყვისაგან *application* ანუ აღჭურვილობა, ხოლო *let* არის კნინობითი სუფიქსი). მართვის კონსოლი ***Windows***-ის კომპონენტია, რომელიც სისტემურ ადმინისტრატორებსა და გამოცდილ მომხმარებლებს მოქნილი ინტერფეისის დახმარებით მოახდინოს სისტემის მუშაობის კონფიგურირება და თვალყურის დევნება. მისი საშუალებით იმართება დრაივერები, ამუშავდება და ჩერდება მოწყობილობები, გამოირთვება უწესივრო მოწყობილობები, აღიქმება დამატებითი ტექნიკური ინფორმაცია.

II.1.4. ბეჭდვის მენეჯერი (*Print Manager*) ადგენს სხვადასხვა დოკუმენტის ბეჭდვის რიგითობასა და პრიორიტეტებს.

■ ***Windows***-ის **<u>სამოსამსახურო გამოყენებითი პროგრამები</u>** განკუთვნილია პერსონალური კომპიუტერის ოპერაციული სისტემის მომსახურებისათვის. ყველა მათგანი შეიძლება ავამუშაოთ მთავარი მენიუდან შემდეგი ბრძანებით:

Start ► *All programs* ► *Standard* ► Services,

(*Старт* ► *Программы* ►*Стандартные* ► *Служебные*).

სამოსამსახურო პროგრამების შემადგენლობაიცვლებაოპერაციულისისტემისვერსიაზედამოკიდებულებით. მაგალითად, ***Windows 7***-ში გვაქვს რამდენიმე სამომსახურო პროგრამა, რომელთაგანაც მოკლედ შევეხებით შემდეგ ორ მათგანს (იხ. ნახ. ***3.1***):

II.2.1. სიმბოლოთა ცხრილი.იგი საშუალებას გვაძლევს ეკრან-ზე დავინახოთ მოცემული შრიფტის ყველა სიმბოლო, დავამყაროთ ურთიერთშესაბამისობა სიმბოლოებსა და კლავიატურის კლავიშებს შო-

რის და შემდგომი მუშაობისათვის ამოვირჩიოთ საჭირო სიმბოლოები.

II.2.2. სისტემის ცნობარისაკმაო კომპაქტური სახით გვაძლევს ინფორმაციას გამოთვლითი სისტემის შესახებ. ეს ცნობები მათი დანიშნულებების შესაბამისად გარკვეულ განყოფილებებადაა დაჯგუფებული. მაგალითად, ჯგუფი ***აპარატურის რესურსები*** მოიცავს ყველა ინფორმაციას კომპიუტერში დაყენებული მოწყობილობების შესახებ, ხოლო ჯგუფი ***პროგრამული გარემო*** აღწერს ოპერაციული სისტემის თითოეული კომპონენტსა და დაყენებული პროგრამებს.

ДИСКЕБИС САСАМСАХУРО ПРОГРАМЕБИС შემადგენლობაც ოპერაციის ცვლილებების შესაბამისად იცვლება ***Windows 7***-ში სხვა არსებობს შემდეგი ორი პროგრამა:

II.3.1დისკის დეფრაგმენტატორი, რომელიც ფრაგმენტაციის აღმოფხვრის გზით ახდენს დისკის მუშაობის ოპტიმიზებას და ამაღლებს დისკში შეღწევადობას. იგი დისკზე არსებულ გრძელ ფაილებს მოკლე ფრაგმენტებისაგან ააწყობს, რითაც მნიშვნელოვნად ჩქარდება მათში შეღწევა და მაღლდება კომპიუტერის მუშაობის ეფექტურობა.

II.3.2. დისკის გამწმენდი განკუთვნილია დისკის იმ ნაწილების გასათავისუფლებად, რომლებიც უკავია უკვე ზედმეტ ფაილებს (ასეთ ფაილებს განსაზღვრავს თავად მომხმარებელი).

6. Windows-ის მეგზური (Explorer)

ფაილური სისტემის მართვისათვის ***Windows***-ში გათვალისწინებულია სპეციალური პროგრამა ***მეგზური*** (*Explorer*), რომელიც საშუალებას გვაძლევს, შევაღწიოთ კომპიუტერის ყველა მოწყობილობაში, დავათვალიეროთ დისკები და კატალოგები და შევასრულოთ სხვადასხვა ოპერაცია კატალოგებსა და ფაილებზე.

არსებობს მეგზურის ამუშავების სამი ხერხი:

1. ამოვირჩიოთ მთავარი მენიუს ბრძანება:

Start►*Allprograms* ► *Standard* ► *Explorer*,

(*Старт* ► *Программы* ►*Стандартные* ► *Проводник*).

2. ნებისმიერკონტეინერზე (სხვა ობიექტების შემცველ ობიექტზე, ე. ი. დისკზე ან საქაღალდეზე) დავაჭიროთ მაუსის მარჯვენა კლავიში და გახსნილ კონტექსტურ მენიუში ამოვირჩიოთ ბრძანება *Open*(*Открыт*);

3. დავაჭიროთ კლავიშების სპეციალურ *WL + E* კომბინაციას.

მეგზურის ფანჯარა ორ პანელადაა გაყოფილი. მარცხენა ნაწილში ჩვეულებრივ გამოსახულია საქაღალდეების ხე, ხოლო მარჯვენა ნაწილში – მიმდინარე საქაღალდის შიგთავსი.

მარცხენა პანელზე განთავსებულია კომპიუტერში დაყენებული დისკური მოწყობილობები და ზოგიერთი სასამსახურო **ფანჯარა (*სამუშაო მაგიდა - Desktop, ჩემი კომპიუტერი My Computer, ჩემი დოკუმენტები - My Document, ქსელური გარემოცვა - My Nrtwork Places სანაგვე ყუთი – Recycrle Bin*)**. ობიექტები გაიხსნება მათზე მაუსის მარცხენა კლავიშის დაჭერით. დისკის გახსნისას ეკრანზე გამოჩნდება მისი საქაღალდეები. საქაღალდეებიც შეგვიძლია გავხსნათ, რის შედეგადაც თვალსაჩინოდ გამოჩნდება ფაილური სისტემის ხისებრი სტრუქტურა.

მეგზურის ***მარცხენა პანელზე*** კონტეინერების (დისკური მოწყობილობებისა და არაცარიელი საქაღალდეების) მარცხნივ მოთავსებულია სამკუთხედი ► (*Windows*7-მდე ვერსიებში სამკუთხედის ნაცვლად გამოყენებული იყო სიმბოლო [+]). ობიექტის გვერდით თუ არავითარი სამკუთხედი არ არის, მაშინ მის შიგნით არ არის არცერთი საქაღალდე, ე.ი. საქაღალდე ცარიელია, ან მასში მარტო ფაილებია. ნიშანი ► ნიშნავს, რომ შიგნით ჩალაგებულია საქაღალდეები. ამ ობიექტზე მაუსით დაწკაპუნებით გავხსნით მოცემულ მოწყობილობას (ან საქაღალდეს) და დავნახავთ ჩალგებული საქაღალდეების მომდევნო დონეს, რომელთა შორის ზოგიერთი მონიშნული იქნება სამკუთხედით.

მეგზურის ***მარჯვენა პანელზე*** აისახება მარცხენა პანელიზე ამორჩეული დისკის (ან საქაღალდის) საქაღალდეები და ფაილები. მარცხენა პანელისაგან განსხვავებით, ობიექტები გაიხსნება მათზე მაუსის მარცხენს კლავიშის ორჯერ დაწკაპუნებით.

მარჯვენა პანელიზე საქაღალდეებისა და ფაილების ასასახად ხუთი რეჟიმია გათვალისწინებული: გვერდების ესკიზი (ვეებერთელა ნიშნები, მსხვილი ნიშნები, ჩვეულებრივი ნიშნები, წვრილი ნიშნები), სია, ცხრილი, ფილა, შიგთავსი. ამ რეჟიმების გადასართველად შეგვიძლია გამოვიყენოთ ***ინსტრუმენტების პანელზე*** არსებული ღილაკი *Additional* (*Дополнительно*).

გახსნილი საქაღალდეები თუ ვერ თავსდება მარცხენა პანელიზე, მაშინ შეგვიძლია გადავანაცვლოთ პანელებს შორისი დამყოფი ზო-

ლი. ამისათვის მასზე ისე უნდა მოვათავსოთ მაუსის მაჩვენებელი, რომ მან ორმხრივ მიმართული ისრის სახე მიიღოს და, მაუსის კლავიშიდან თითის აუშვებლად გამყოფი ზოლი ახალ ადგილზე გადავათრიოთ.

სამუშაო მაგიდის ორივე მხარეზე არსებული დაბრუნების ხაზებით შეგვიძლია „გადავფურცლოთ“ ფაილური სისტემა.

მეგზური ობიექტებზე ყველა საჭირო ოპერაციის შესრულების საშუალებას გვაძლევს.

■ ***საქაღალდის შესაქმნელად საჭიროა:***

1. დავაჭიროთ ინსტრუმენტების პანელზე არსებულ ღილაკს *New Folder*(*ახალი საქაღალდე*) ან შევასრულოთ კონტექსტური მენიუს ბრძანება:

New ► *Folder;* (*Создать*► *Папку*);

ამისათვის მაუსის მარჯვენა ღილაკი დავაჭიროთ ფანჯრის თავისუფალ ადგილზე, გამოსულ კონტექსტურ მენიუში მაუსის მარცხენა ღილაკით დავაჭიროთ ბრძანებას *New* (Создать), ხოლო შემდეგ გამოსულ კონტექსტურ მენიუში ავირჩიოთ ბრძანება*Folder* (*Папка*)**;**

2. ფანჯარაში გაჩენილ საქაღალდეზე არსებული წარწერა *„New Folder“* კლავიატურით შევცვალოთ ჩვენთვის სასურველი სახელით და დავადასტუროთ იგი კლავიშ *ENTER*–ზე ზემოქმედებით;

■ ***ტექსტური დოკუმენტის შესაქმნელად საჭიროა:***

1. შევასრულოთ კონტექსტური მენიუს ბრძანება:

New ► *TextDocument;*(*Создать*►*Текстовыйдокумент*);

2. დავარქვათ ეკრანზე გამოჩენილ ფაილს სახელი დადავადასტუროთ იგი კლავიშ *Enter*–ზედაჭერით;

3. ფაილის სახელზე მაუსის მარცხენა ღილაკის ორმაგი დაჭერით გავხსნათ ფაილი;

4. ავკრიფოთ ფაილის ტექსტი;

5. შევასრულოთ მენიუს ბრძანება *File* ► *Save*(*Файл* ► *Сохранить*);

6. დახუროთ ფაილი.

■ ***ობიექტების კოპირებისათვის საჭიროა:***

1. მაუსის მარცხენა ღილაკის დაჭერით გამოვყოთ ობიექტი (საქაღალდე ან ფაილი) (ობიექტების ჯგუფი გამოიყოფა *Ctrl*კლავიშის დაჭერილ მდგომარეობის დროს მაუსის მარცხენა კლავიშის დაწკაპუნებით);

2. შევასრულოთ კონტექსტური მენიუს *Copy* (*Копировать*) ბრძანება ან გამოვიყენოთ კლავიშების სტანდარტული კომბინაცია *Ctrl + C*;

3. გავხსნათ დისკი ან ობიექტის გადასაკოპირებელი საქაღალდე;

4. შევარულოთ კონტექსტური მენიუს ბრძანება*Paste*(*Вставить*), ან გამოვიყენოთ კლავიშების სტანდარტული კომბინაცია *Ctrl + V*;

■ ***ობიექტის გადასაადგილებლად საჭიროა:***

1. გამოვყოთ ობიექტი (ან ობიექტების ჯგუფი);

2. შევასრულოთ კონტექსტური მენიუს ბრძანება *Cut*(*Вырезать*)ან გამოვიყენოთ კლავიშების სტანდარტული კომბინაცია *Ctrl + X*;

3. გავხსნათ დისკი ან ობიექტის გადასაკოპირებელი საქაღალდე;

4. შევასრულოთ კონტექსტური მენიუს ბრძანება *Paste* (*Вставить*), ან გამოვიყენოთ კლავიშების სტანდარტული კომბინაცია *Ctrl +V*;

კოპირებისა და გადადგილებისათვის მოსახერხებელია გამოვიყენოთ მაუსი. ამისათვის ამორჩეულ ობიექტზე დავაჭიროთ მაუსის მარცხენა კლავიში და ამ ღილაკიდან თითის აუშვებლად გადავაადგილოთ მაუსი იმ საქაღალდეზე ან დისკზე, სადაც გვინდა გადავიტანოთ ობიექტი. დაჭერილი ***Ctrl*** კლავიშის დროს მაუსით გადათრევა გადააკოპირებს ამორჩეულ ობიექტს;

■ ***ობიექტის სახელის შესაცვლელად საჭიროა:***

1. გამოვყოთ ობიექტი (საქაღალდე ან ფაილი);

2. შევასრულოთ კონტექსტური მენიუს ბრძანება *Rename* (*Переименовать*)ან გამოვიყენოთ კლავიში *F2*;

3. შევცვალოთ ობიექტის სახელი და დავაჭიროთ კლავიშს *Enter*.

■***ობიექტის გასაძევებლად საჭიროა:***

1. მოვნიშნოთ ობიექტი (ან ობიექტების ჯგუფი);

2. შევასრულოთ კონტექსტური მენიუს ბრძანება ***Delete*** (*Удалить*)ან ვისარგებლოთ*F2*კლავიშით.;

3. დავადასტუროთ ობიექტის გაძევება ამის შესახებ დასმულ შეკითხვაზე დადებითი პასუხის გაცემის გზით.

3.3. გამოყენებითი პროგრამული უზრუნველყოფის სახეები

თანამედროვე კომპიუტერის გამოყენებით შესაძლებელია ერთმანეთისაგან თვისობრივად განსხვავებული უამრავი სხვადასხვა ამოცანის გადაწყვეტა, რისთვისაც დამუშავებული იქნა გამოყენებითი პროგრამა. მათი კლასიფიკაცია და სახეები ***3.2*** ცხრილშია მოყვანილი. მიკროელექტრონიკის სფეროში არსებულმა მიღწევებმა სტიმული მისცა ახალი გამოყენებითი პროგრამების დამუშავებას. მათი სტრუქტურა და აგების პრინციპები დამოკიდებულია ოპერაციულ სისტემაზე, რომლის ჩარჩოებში მუშაობს ისინი.გამოყენებითი პროგრამების სიმრავლე იყოფა ორ ჯგუფად: პრობლემურად ორიენტირებულ პროგრამებად და ინტეგრირებულ პაკეტებად, ხოლო ეს უკანასკნელი

იყოფა სრულად შეკრულ და ობიექტურად შეკრულ ინტეგრირებულ პაკეტებად.

■ ***პრობლემურად ორიენტირებული პროგრამები*** ყველაზე მრავალრიცხოვანია. დღეისათვის ასეული ათასი ასეთი პროგრამაა დამუშავებული.***3.2*** ცხრილში მხოლოდ უმნიშვნელო რაოდენობის ასეთი პროგრამებია მოყვანილი. ***საქმისწარმოებაში*** ყველაზე ხშირად გამოიყენება ტექსტური რედაქტორები და ცხრილური პროცესორები. ***სავაროდ გამოსვლებისათვის*** წარმატებით გამოიყენება პრეზენტაციათა მოსამზადებელი და გრაფიკული პროგრამები, ხოლო ***ეკონომისტებისათვის*** ფასდაუდებელია ეკონომიკური დანიშნულების პროგრამები, წარმოების ავტომატიზებულად მართვის სისტემები, სტატისტიკურმათემატიკური პროგრამები და ა.შ.

ბოლო წლებში შეიმჩნევა ინტეგრირებული პაკეტების მასობრივად გამოყენების ტენდენცია. ***ინტეგრირებული პაკეტები*** ეწოდება ფუნქციურად ერთმანეთის შემავსებელი და საერთო ტექნოლოგიის გამომყენებელი რამდენიმე პროგრამული პროდუქტის ერთობლიობას. განასხვავებენ ***სრულად შეკრულ*** და ***ობიექტურად შეკრულ*** ინტეგრირებული პაკეტებს (იხ.ცხრ. ***3.2***).

■ ***სრულად შეკრული ინტეგრირებული*** პაკეტი მრავალფუნქციური ავტონომიური კომპლექსია. ამ პაკეტის ჩარჩოებში მონაცემებს შორის სრული კავშირია უზრუნველყოფილი, რის გამოც განცალკევებულ ანალოგურ სპეციალიზებულ პროგრამებთან შედარებით ვიწროვდება თითოეული პროგრამის შესაძლებლობა.

■ ***ობიექტურად შეკრული ინტეგრირებული*** პაკეტი (იხ. ცხრ. 3.2) სპეციალიზებულ პროგრამებს საერთო რესურსული ბაზის ჩარჩოებში ობიექტების დონეზე აერთიანებს***OLE*** ტექნოლოგია (იხ. გვ. ***102***).ასეთ პაკეტებში ინტერფეისების ურთიერთშეთანხმებულობას განაპირობებს ერთნაირი სადიალოგო ფანჯრების, პიქტოგრამებისა და მენიუს გამოყენება. სრულად შეკრული ინტეგრირებული პაკეტისაგან განსხვავებით ერთ პაკეტში შემავალი პროგრამების ფუნქციური ფუნქციური შესაძლებლობა არ ვიწროვდება, და, გარდაამისა, ისინი გაცილებით (***20-50%***-ით) იაფია. ობიექტურად შეკრულიპაკეტებია (იხ. ცხრ. ***3.2***): *Microsoft Office OpenOffice.org; Borland Office* და *Lotus SmartSuite*; ძალზე პოპულარულია ობიექტურად ინტეგრირებული ***Mikrosoft Offise*** პაკეტი**.** ამას განაპირობებს ***Mikrosoft***–ის მიერ გატარებული აგრესიული მარკენტიკული პოლიტიკა და პაკეტის ღირსებები. კერძოდ, მას აქვს მოსახერხებელი ინტერფეისი, კორპორაციულ ქსელებში დოკუმენტების კოლექტიური დამუშავების გაუმჯობესებული შესაძლებლობები, განვითარებული საცნობარო სისტემა, მოსახერხებელი და მარტივი ინტელექტუალური დანართები, სპეციალური დამხმარე

პროგრამა (***Offise Assistant***) და, ბოლოს, იგი უზრუნველყოფს საჭირო დოკუმენტებთან მომხმარებლის სწრაფ შესვლას.

ცხრ. 3.2. გამოყენებითი პროგრამული უზრუნველყოფის კლასიფიკაცია

1. პრობლემურად ორიენტირებული პროგრამები	■ ***ტექსტური რედაქტორები*** (*Microsoft Word, OpenOffice.org Writer, Corel WordPerfect, Ami Pro, «Lexicon»და სხვ.*); ■ ***ცხრილური პროცესორები***(*Microsoft Excel, OpenOffice.org Calc, Quattro Pro, Super Calc, Lotus 1-2-3 და სხვ*); ■ ***მონაცემთა ბაზების მართვის სისტემები***(*Microsoft Access, OpenOffice.org Base , FoxPro, Oracle, Paradox და სხვ.)*; ■ ***გრაფიკის დამუშავების სისტემები*** (*Corel Draw, Adobe PhotoShop, OpenOffice.org Draw, Macromedia FreeHand, Aldus PhotoStyler და სხვ.* ■ ***ვიდეორედაქტორები*** (*Adobe Premiere, Video Craft, Maya და სხვ*); ■ ***ბგერის რედაქტორები***(*SoundForge, GoldWave, AWaveდა სხვ*); ■ ***სადემონსტრაციო გრაფიკის რედაქტორები***(*Microsoft PowerPoint, OpenOffice.org Impress, Freelance Graphics, Harvard Graphics და სხვ*); ■ ***სიმბოლოთა ამოცნობის სისტემები*** (*Abbyy FineReader, CuneiForm, OmniPage და სხვ*); ■ ***სააგამომცემლო სისტემები*** (*Microsoft Publisher, Adobe PageMa-ker, Quark XPress, Corel Ventura და სხვ*); ■ ***საბუღალტრო და საფინანსო პროგრამები***; ■ ***სამართებლივი მონაცემთა ბაზები***; ■ ***დაპროექტების ავტომატიზების სისტემები*** (*Autodesk AutoCad, DesignCAD, Drawbaze, UltimateCAD და სხვ*); ■ ***მათემატიკური გამოთვლების სისტემები*** (*OpenOffice.org Math, Maple, Mathematica MathCAD, MathLab და სხვ*); ■ ***მონაცემების მათემატიკური ანალიზის სისტემები სისტემები*** (*Microsta, SPSS, Statgraph, Statistica და სხვ*); ■ ***ცნობარები და ენციკლოპედიები, მათ შორის მულტიმედიური;*** ■ ***სხვადასხვა თამაშების პროგრამები***	
2. ინტეგრირებული პაკეტები	***2.1. სრულად შეკრული ინტეგრირებული პაკეტები***	■ ***FrameWork;*** ■ ***Symphony;*** ■ ***Microsoft Works;*** ■ ***Lotus Work;***
	2.2. ობიექტურად შეკრული ინტეგრირებული პაკეტები	■ ***Microsoft Office;*** ■ ***OpenOffice.org;*** ■ ***Borland Office;*** ■ ***Lotus SmartSuite;***

Mikrosoft Offise პაკეტის ორგანიზებისათვის დამუშავებულია: ***1)*** ტექსტური რედაქტორი ***Word***; ***2)*** ცხრილური პროცესორი ***Exsel***; ***3)*** საპრეზენტაციო პროგრამა ***PowerPoint***; ***4)*** სწრაფი ჩანაწერების გასაკეთებელი პროგრამა ***OneNote*** (*ბლოკნოტი*); ***5)*** საფოსტო კლიენტის ფუნქციების მქონე პერსონალური პროფესიული საინფორმაციო მენეჯერი ***Outlook***; ***6)*** საგამომცემლო სისტემა ***Publisher***; ***8)*** მონაცემთა ბაზის მართვის სისტემა ***Access***; ***7)*** მონაცემების ფორმების შესაქმნელშესარები პროგრამა ***InfoPath***; ***8)*** ქსელური პროგრამა ***Lync***; ***9)*** დოკუმენტების მართვისა და შენახვის ***Microsoft Share Point*** ჯგუფის საიტებთან შეუფერხებლად შესაღწევი პროგრამა ***Share Point Workspace***; ***10)*** ბიზნესის დამგეგმავი პროგრამა ***SharePoint Workspace***; ***10)*** ბიზნესის დამგეგმავი პროგრამა ***Project*** და ***11)*** ვექტორული გრაფიკის ***Visio*** პროგრამა.

ზემოთ ჩამოთვლილი კომპონენტების დაჯგუფების გზით ***Mikrosoft*** უშვებს ***Mikrosoft Offise***-ად წოდებული შემდეგ ხუთ პაკეტს:

1)საოჯახო-სასწავლო B_1 ***პაკეტი*** რომელშიც***Word,Excel, Power-Point*** და ***OneNote***პროგრამებია ინტეგრირებული;

2)ოჯახისა და ბიზნესის $B_2 = B_1 \cup$***Outlook პაკეტი;*** სადაც ***Outlook*** *პერსონალური საინფორმაციო მენეჯერია;*

3)სტანდარტული $B_3 = B_2 \cup$***Publisher პაკეტი,*** სადაც ***Publisher*** *სამაგიდო საგამომცემლო სისტემაა;*

4) ***პროფესიული*** $B_4 = B_3 \cup$***Access პაკეტი***, სადაც ***Access*** *მონაცემთა ბაზის მართვის რელაციური სისტემაა;*

5)პროფესიული პლუს $B_5 = B_4 \cup$(***InfoPath***&***Lync &SharePoint Workspace***) ***პაკეტი,*** სადაც ***InfoPath*** არის XML-ის საფუძველზე მონაცემების შეტანის ფორმების დამუშავებისათვის გამოყენებული პროგრამა, ***Lync –*** ვიდეოკონფერენციის სისტემა, ხოლო ***SharePointWorkspace*** – საიტების დამუშავებისათვის გამოსაყენებელი პროგრამაა.

IV თავი
ტექსტის დამუშავების ტექნოლოგიები

4.1. ტექსტური რედაქტორები და პროცესორები

ტექსტის დამუშავების ტექნოლოგიები საინფორმაციო ტექნოლოგიების ერთ-ერთი ყველაზე მასობრივი სახეა. დამწერლობის შექმნამ შესაძლებლობა მოგვცა დროსა და სივრცეში გავრცელების მიზნით ინფორმაცია შეგვენახა რაიმე სახის მატერიალურ ობიექტზე, ე. ი ეს უკანასკნელი გარდაგვექმნა ინფორმაციის მატარებლად ანუ მზიდად. ამან საფუძველი დაუდო ***დოკუმენტირებული ინფორმაციის წარმოშობას***. სიტყვა ***„დოკუმენტი“*** წარმოქმნილია ლათინური ***„documentum“***-ისაგან, რაც ***მოწმობას*** ნიშნავს. დოკუმენტზე, როგორც მოწმობაზე, საუბრისას თვალწინ, უპირველეს ყოვლისა, წარმოგვიდგება ქაღალდზე დატანილი გარკვეული ტექსტი, რომელიც ხელმოწერითა და ბეჭდით არის დამოწმებული, მაგრამ არის დოკუმენტის კიდევ ერთი ნაირსახეობა, ე. წ. ***რეტროსპექტული*** (*წარსულისაკენ მიმართული*) დოკუმენტი, რომელსაც***ისტორიული დოკუმენტი*** ეწოდება. ისტორიული დოკუმენტებია რაიმე ისტორიული მოვლენის, პირის, ეპოქის არსებობის დამადასტურებელი მატეანეები, ქრონიკები, ჩანაწერები.

ტექსტების შემქნელ და დამამუშავებელ ძირითად ინსტრუმენტებად პერსონალური კომპიუტერების გადაქცევის შემდეგ ტერმინმა ***„დოკუმენტი“*** კიდევ ერთი მნიშვნელობა შეიძინა. კერძოდ, შემოვიდა ***„ტექსტური დოკუმენტის“*** ცნება, რომელითაც აღინიშნება ***კომპიუტერზე შექმნილი და ფაილში შენახული ტექსტი***. კომპიუტერზე შექმნილიდოკუმენტი, გარდა ტექსტისა, შეიძლება შეიცავდეს ფორმულებს, დიაგრამებს, ნახატებს, ცხრილებს, კოლონტიტულებს (ტექსტის ზევით ან ქვემოთ არსებულ არეებს, ე.წ. ზედა და ქვედა კოლონტიტულებს, რომლებშიც შეიძლება აისახოს ნებისმიერი ინფორმაცია) და ა. შ. ტექსტში შეიძლება მრავაფეროვანი შრიფტები იყოს გამოყენებული, იცვლებოდეს ველების ზომები.

ასეთ დოკუმენტს ***გარკვეული სახით გაფორმებულ დოკუმენტს*** უწოდებენ. ყოველდღიურ ცხოვრებაში ჩვენ გვიხდება გამოვიყენოთ ისეთი სხვადასხვა სახის დოკუმენტი, როგორიცაა განცხადებები, ბრძანებები, ინსტრუქციები, სტატიები, მოთხრობები, ლექსები და ა. შ. დოკუმენტების შექმნისა და დამუშავების მიზნით შექმნილ გამოყენებით პროგრამებს უწოდებენ ***ტექსტურ რედაქტორებსა*** და ***ტექსტურ პროცესორებს***.

ტექსტური რედაქტორი და ტექსტური პროცესორი ერთმანეთისაგან განსხვავდება ფუნქციური შესაძლებლობებით. ***ტექსტურ რედაქტორებში რეალიზებულია ტექსტების შექმნისა და რედაქტირების საბაზისო ფუნქციები:*** შეტანა, კოპირება, ადგილის გამოცვლა, გაძევება, ტექსტის ფრაგმენტების მოძებნა და შეცვლა, დოკუმენტის შენახვა გარე მეხსიერებაზე, დოკუმენტის დაბეჭდვა. ამგვარად, ტექსტური რედაქტორი საშუალებას გვაძლევს ყოველგვარი გაფორმების გარეშე უბრალოდ „ავკრიფოთ“ გარკვეული ტექსტი. ყოველგვარი გაფორმებისაგან თავისუფალ ტექსტს უწოდებენ ***„plain text“***-ს (მარტო ტექსტს, „ბრტყელ“ ტექსტს, მარტივ ტექსტს). დღეს გავრცელებული ტექსტური რედაქტორებია ოპერაციულ სისტემა ***Windows***-ში არსებული ***ბლოკნოტი***(***Notepad***), აგრეთვე ***Linux***-ში არსებული ***Vi*** და ***Emacs.*** მათ გარდა არსებობს შემდეგი სახის ტექსტური რედაქტორები: *Emacs, jEdit, Kate, Vim, GNUnano, EditPlus, EmEditor, SciTE, NEdit, Notepad++ (GNUGPL), NotepadGNU , Oiynsoft Text Editor, PSPad , RJ TextEd, TEA, Crimson Editor, AkelPad , UltraEdit, TextEdit, VEdit , DPAD, Rnote.*

ტექსტური პროცესორებს საბაზისო ფუნქციებზე გაცილებით მეტი ფუნქციების შესრულება შეუძლია. ისინი დამატებით საშუალებას გვაძლევს: ა) ავტომატიზებულად შევადგინოთ ალფაბეტური და საგნობრივი საძიებლები, სარჩევები, სიები; ბ) ავტომატიზებულად ვმართოთ მეხსიერება; გ) დავაფორმატოთ ტექსტი.

დაფორმატება განსაზღვრავს ტექსტის ***გარეგნობას*** და არა მის ***შინაარსს***, თუმცა ისინი ხშირად ერთმანეთთანაა დაკავშირებული. მაგალითად, ვინმეს სახელზე ოფიციალური განცხადების შედგენისას ადრესატის რეკვიზიტები მარჯვენა ზედა კუთხეში ჩაიწერება. მრავალ დოკუმენტს ცხრილის სახე აქვს. დოკუმენტის განყოფილებათა სათაურები ხაზგასმით ან მსხვილი შრიფტით იწერება და სტრიქონის შუაში განთავსდება.

ეკრანზე ან დასაბეჭდად გამოტანის დროს დაფორმატებულ ტექსტიანი ფაილი შეიცავს როგორც ტექსტის შინაარსს, ისე მისი დაფორმატების მონაცემებს.

Microsoft Office პაკეტის შემადგენლობაში შედის ტექსტური ***Microsoft Word***პროცესორი. ბოლო წლებში სულ უფრო და უფრო პოპულარული ხდება უნივერსალური საოფისე ***OpenOffice.org*** პაკეტის შემადგენლობაში არსებული ტექსტური ***OpenOffice.org Writer*** პროცესორი, რომელიც სხვადასხვა ოპერაციულ პლატფორმზე მუშაობს. ეს პაკეტი მიეკუთვნება თავისუფლად გავრცელებად პროგრამულ უზრუნველყოფას. დანარჩენი პოპულარული ტექსტური პროცესორების სია ასეთია: *AbiWord, Adobe InCopy, Apple iWork Pages, ChiWriter, JWPce , LaTeX, LibreOffice Writer, Lotus WordPro,Microsoft Works, PolyEdit, WordPad, WordPerfect.*

ტექსური პროცესორების განვითარება მიმართულია მათი ინტელექტუალური შესაძლებლობების ამაღლებისაკენ, კერძოდ, იმისაკენ. რომ მათ შეეძლოს ავტომატურად შეამოწმოს მართლწერა, გადათარგმნოს ტექსტი ერთი ენიდან მეორეზე და ა. შ.

1. მართლწერის შემოწმება.

მართლწერის შემოწმება ნიშნავს ორთოგრაფიის, გრამატიკისა და სტილისტიკის შემოწმებას. შეიძლება შემოწმდეს მხოლოდ ტექსტური პროცესორისათვის მისაღებ ერთ-ერთ ენაზე დაწერილი ტექსტი. ყველაზე ხშირად ტექსტის ენას თავად ტექსტური პროცესორი შეირჩევს.

<u>ორთოგრაფიის შემოწმებისათვის</u>უნდა არსებობდეს ***ჩაშენებული ლექსიკონები.*** ორთოგრაფიული შემოწმების დროს ტექსტური დოკუმენტის თითოეული სიტყვა უდარდება ლექსიკონში არსებულ სიტყვებს. უფრო ზუსტად, სიტყვაში შემავალ შესამოწმებელ სიმბოლოთოთა კოდები უდარდება აღნიშნული სიმბოლოს ლექსიკონში ჩაწერის ნიმუშს. ამის გამო წარმოიშობა სიმბოლოთა კოდების შეთავსებადობის პრობლემა. სხვადასხვა კომპიუტერებზე შექმნილ დოკუმენტებში ორთოგრაფიული შემოწმების სწორად ჩატარებისათვის საჭიროა არსებობდეს ეროვნული სიმბოლოების კოდირების ერთიანი სისტემა. ამ პრობლემას წყვეტს კოდირების საერთაშორისო კოდის – ***Unicode***-ის (***უნიკოდის***) გამოყენება, რომლის სტანდარტი ***1991*** წელს იქნა მიღებული. ***16***-თანრიგიანი უნიკოდი $2^{16}=65536$ სიმბოლოს კოდირების საშუალებას გვაძლევს. ***16***-თანრიგიანი უნიკო-

დებით შეიძლება ინგლისური (ლათინური), რუსული (კირილიცა), ბერძნული ასოების, ჩინური იეროგლაფების, მათემატიკური სიმბოლოებისა და მრავალი სხვა ეროვნულ ალფაბეტთა ასოების კოდირება.

როგორი უნდა იყოს ლექსიკონის მოცულობა, როგორ უნდა განვასზღვროთ მისი ოპტიმალური ზომება? ამ კითხვაზე ცალსახა პასუხი არ არსებობს. ლექსიკონის მოცულობის გაზრდა არ გვაძლევს შესამოწმებელ ტექსტში შესაძლო შეცდომების არარსებობის გარანტიას. უფრო მნიშვნელოვანია ანალიზისა და შეცდომების გამოაშკარავების ალგორითმი. დოკუმენტის ტექსტი თუ შეიცავს ლექსიკონებში არარსებულ სიტყვას, ტექსტური პროცესორი მას ტექსტში აღმოჩენილ ორთოგრაფიულ შეცდომად მიიჩნევს. სამაგიეროდ იგი სწორად ჩათვლის დოკუმენტში ორთოგრაფიული თვალსაზრისით სწორად დაწერილ, მაგრამ კონტექსტიდან ამოვარდნილ სიტყვას; მისთვის სწორია, მაგალითად, ისეთი ტექსტები, როგორიცაა „ვარდისებური ყვავი“, „ვარდისებური ეკალი“, „ვარდისებური ბეჰემოტი“.

ტექსტური პროცესორები მომხმარებელს საშუალებას აძლევს შექმნას საკუთარი ლექსიკონები, არსებულ ლექსიკონებს დაუმატოს ან გამოაკლოს სიტყვები. ***სამომხმარებლო ლექსიკონები*** გამოიყენება ძირითად ლექსიკონში არარსებული სწორად დაწერილი სიტყვების შესანახად. ისინი ყველაზე უფრო ხშირად იქმნება ვიწროსპეციალურ, მაგალითად, ტექნიკურ ან სამეცნიერო ტექსტებთან მუშაობისათვის. ტექსტურ ***Microsoft Word*** პროცესორში არსებობს ძირითადი ლექსიკონი ***CUSTOM.dic***. ყველა სამომხმარებლო ლექსიკონი და ძირითადი ლექსიკონი უსიტყვოდ შეინახება საქაღალდეში „***Documents and Settings***“. ლექსიკონების ფაილების გაფართოებაა ***dic***, მაგრამ ისინი ჩვეულებრივი ტექსტური ფაილებია და მათთან შეიძლება, მაგალითად ვიმუშაოთ ტექსტურ რედაქტორ ***ბლოკნოტში*** (***Notepad***).

<u>გრამატიკის შემოწმებისას</u> გამოაშკარავდება: წინდებულების არასწორად გამოყენება, წინადადებებში სიტყვების შეუთანხმებლობა და ა. შ. გრამატიკული შემოწმება ხორციელდება წესების ფიქსირებული ნაკრების საფუძველზე. ნაკრების შემადგენლობა შეგვიძლია საჭიროებისამებრ ვცვალოთ მასში გარკვეული წესების შეტანის ან გამორიცხვის გზით.

<u>სტილისტიკის შემოწმება</u>საშუალებას გვაძლევს დოკუმენტში აღმოვაჩინოთ არალიტერატურული და დიალექტური სიტყვები და გამოთქმები.

მართლწერის (ორთოგრაფიის, გრამატიკის, სტილისტიკის) შემოწმების პარამეტრები შეგვიძლია ვმართოთ (გავმართოთ). მაგალითად, ტექსტურ რედაქტორ *Microsoft Word*-ში ეს შესაძლებელია თუ შევასრულებთ ბრძანებას *File* ►*Word Option* (*Файл* ► *Параметры*) და ამოვირჩევთ პუნქტს *Proofing* (*Правописание*).უმეტესი შემთხვევებისათვის მართლწერის შემოწმების პარამეტრები უსიტყვოდ ოპტიმალურადაა დაყენებული.

2. ავტომატური თარგმნა

თანამედროვე ტექსტური პროცესორები ტექსტის ერთი ენიდან მეორე ენაზე თარგმნის პროცესის ავტომატიზების საშუალებას იძლევა. კერძოდ, ინტერნეტთან კომპიუტერის მიერთების შემთხვევაში პროგრამა *World-Lingvo*-ს დახმარებით ტექსტური პროცესორი *Microsoft Word* ცალკეული სიტყვებისა და ფრაზების თარგმნის საშუალებასაც გვაძლევს. ამისათვის საჭიროა ***მოვინიშნოთ*** სიტყვა ან ფრაზა, გავხსნათ მისი ***კონტექსტური მენიუ*** და ვისარგებლოთ ამ მენიუს ბრძანებით *Translate* (*язык*).

ტექსტის ერთი ენიდან მეორე ენაზე თარგმნის პროცესის ავტომატიზების ამოცანა ძალიან რთულია, რადგან თარგმნის დროს საჭიროა არა მარტო სწორად შეირჩეს სიტყვა, არამედ გათვალისწინებული იყოს ენების გრამატიკული და კულტურული თავისებურებებიც. თარგმნის ავტომატიზებისათვის გამოიყენება ***პროგრამა-ლექსიკონები*** **Abby Lingvo** (შემქმნელი ფირმა: ***Abby***), **Multilex** (შემქმნელი ფირმა: *МедиаЛингва*)და ***მთარგმნელი პროგრამები***, როგორიცაა, მაგალითად, **Promt** (შემქმნელი ფირმა: *ПРОулеМТ*); მათი საშუალებით შეგვიძლია ვთარგმნოთ დოკუმენტის მთელი ტექსტი და არა ცალკეული სიტყვები.

მთარგმნელი პროგრამები ელექტრონულ ლექსიკონებს დამხმარე კომპონენტებად იყენებს. ლექსიკონები შეიძლება თემატურ ჯგუფებად დავყოთ. მთარგმნელი პროგრამების შემადგენლობაში შემავალი ***ინტელექტუალური ანალიზატორი*** თარგმნის პროცესში ავტომატურად ადგენს ტექსტის თემატიკას და სიტყვათა მნიშვნელობებს შესაბამისი ჯგუფის ლექსიკონების დახმარებით განსაზღვრავს; ეს მნიშვნელოვნად ამაღლებს თარგმნის ხარისხს. მთარგმნელების შემადგენ-

ლობაში შეგვიძლია ჩავრთოთ ტექსტური რედაქტორები. ისინი საშუალებას გვაძლევს შევიტანოთ საწყისი ტექსტები, მოვახდინოთ თარგმანის რედაქტირება, შევინახოთ თარგმნის შედეგები, ტექსტები გადავაგზავნოთ ელექტრონული ფოსტით, ტექსტები ვთარგმნოთ„ფურცლიდან“, ე. ი. სკანერირებული ტექსტები.

3. სინონიმების ლექსიკონი და თეზაურუსები

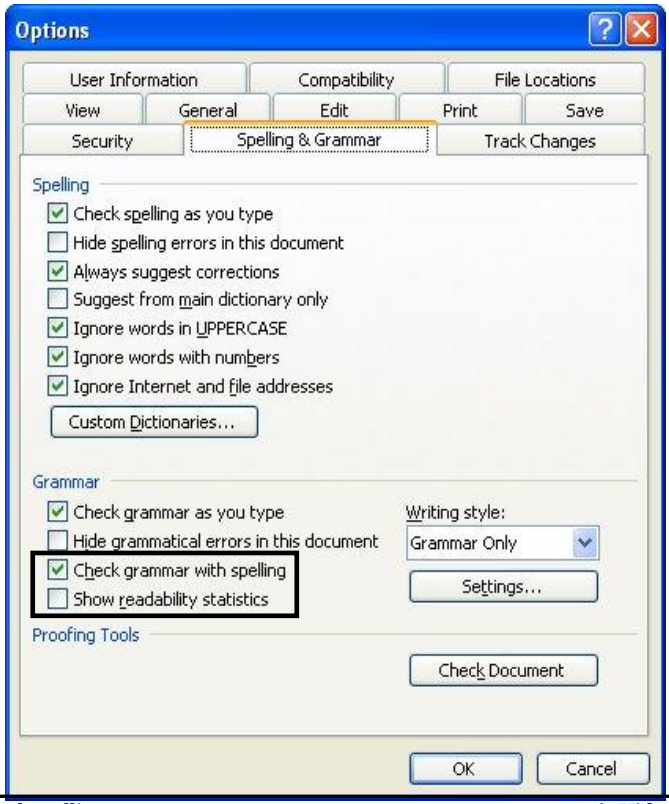
ტექსტური დოკუმენტის შექმნისას სასურველი არ არის ერთი და იგივე სიტყვების გამეორება, დაშტამპული გამოთქმებისა და ჟარგონების გამოყენება. ამ ამოცანის გადაწყვეტაში გვეხმარება სინონიმების გამოყენება. ტექსტურ პროგრამა *Microsoft Word*-ში რომელიმე სიტყვისათვის სინონიმების შესარჩევად საჭიროა ***მოვნიშნოთ*** იგი, გავხსნათ ამ სიტყვის ***კონტექსტური მენიუ*** და ავირჩიოთ ოპცია *Synonyms* (*Синонимы*). მონიშნულ სიტყვას შეიძლება ჰქონდეს რამდენიმე სინონიმი, ან საერთოდ არ ჰქონდეს იგი.

სინონიმებისა და ანტონიმების სრული სიის, სიტყვის სხვადასხვა მნიშვნელობათა ნუსხის, აგრეთვე და ურთიერთდაკავშირებულ სიტყვათა ჩამონათვალის გადასათვალიერებლად უნდა ვისარგებლოთ მონიშნული სიტყვის ***კონტექსტური მენიუს*** პუნქტით *Thesaurus*(*Тезаурус*). ტერმინი ***„თეზაურსი“*** ნაწარმოებია ბერძნული სიტყვისაგან ***„thesaurus“***, რაც ითარგმნება როგორც ***„საგანძური“***. თეზაურსი განსაკუთრებული სახის ლექსიკონია, რომელშიც სიტყვები ერთმანეთთან ლექსიკურ მიმართებათა საფუძველზეა დაკავშირებული; მაგალითად, ისინი შეიძლება წარმოადგენდეს სინონიმებს ან ანტონიმებს. ასეთი სახის ლექსიკონი, ***უპირველეს ყოვლისა***, მეტყველი და «ძარღვიანი»ენით შედგენილი დოკუმენტის მომზადებაში გვეხმარება, მაგრამ ***უფრო მნიშვნელოვანია*** ის, რომ, ვინაიდან თეზაურუსი სიტყვების აზრს სხვა სიტყვებთან თანაფარდობის საშუალებით ასახავს, ამიტომ იგი შეიძლება ***ხელოვნური ინტელექტის სისტემებში***გამოვიყენოთ.

4. ადვილწაკითხვადობის (***readability***) *შემოწმება*

ტექსტურ პროცესორთა პროგრამების პოპულარულ პაკეტებში ჩაშენებულია კითხვადობის დონის ტესტირებისათვის

განკუთვნილისპეციალურიინსტრუმენტები. ისინიშეფასებისზომის ერთეულადიყენებს*„ადვილსაკითხაობის(**readability**) ფლეშ-კინკეიდისეულინდექსს“*. ინტერნეტის ანალიზმა გვიჩვენა, რომ ***Word***-ს



1. **Check spelling as you type** – ავტომატურად შეამოწმე ორთოგრაფია; 2**. Hide spelling errors in this document –** დოკუმენტში დამალე მართლწერის შეცდომები; 3. ***Alwa ys suggest corrections***- ყოველთვის შემოგვთავაზეთ შესწორებები; 4. ***Suggest from main dictionari only*** – მხოლოდ მთავარი ლექსიკონიდან შემოგვთავაზეთ; 5. ***Ignore words in UPPERCASE*** –უგულველყავით მთავრული ასოებით შედგენილი სიტყვები; 6. ***Ignore words with numbers*** – უგულვებელყავით რიცხვებიანი სიტყვები; 7. **Ignore Internet and file addresse** – უგულველყავით ფაილის მისამართები და ინტერნეტი; 8. ***Check grammar as you type*** **–** შეამოწმეთ დაწერილის გრამატიკა; 9. ***chec k grammar with spelling***-შეუხამეთ მართლწერა გრამატიკას;10. ***Show readability statistics*** – გვაჩვენეთ ადვისაკითხაობის სტატისტიკა.

ნახ. 4.1. მართლწერის აწყობისათვის განკუთვნილი *ფურცელი Proofing*

შეუძლია შეაგროვოს ამ ინდექსის გამოსათვლელად საჭირო მონაცემები.

MicrosoftWord-ის მიერმ ართლწერის შემოწმების შემდეგ შეგვიძლია ეკრანზე გამოვიტანოთ ცნობები დოკუმენტის ადვილწაკითხვადობის შესახებ, რომელშიც ჩართული იქნება ***ფლეშის ტესტის*** ან ***ფლეშ-კინკეიდის სასკოლო ტესტის*** ეშვეობით გამოთვლილი მაჩვენებლებიც.

სტატისტიკის გამოტანის ოფცია უსიტყვო შეთანხმებით გამოთიშულია. მის ჩასართავად:

■ *Word*-ში უნდა გავხსნათ ჩანართი ***File*** [*Файл*] და ამოვირჩიოთ ***word Option*** (*Параметры*) პუნქტი;

■ ამოვირჩიოთ პუნქტი ***Proofing*** (*Правописание*), რომლის შედეგადაც გაიხსნება ***4.1*** ნახაზზე მოყვანილი ფანჯარა; დავრწმუნდეთ, რომ ალამი დაყენებულია ამ ფანჯრის**«*check grammar with spelling*»**(«*Вовремяпроверкиорфографиитакжепроверятьграмматику*»)ოფცია ში;

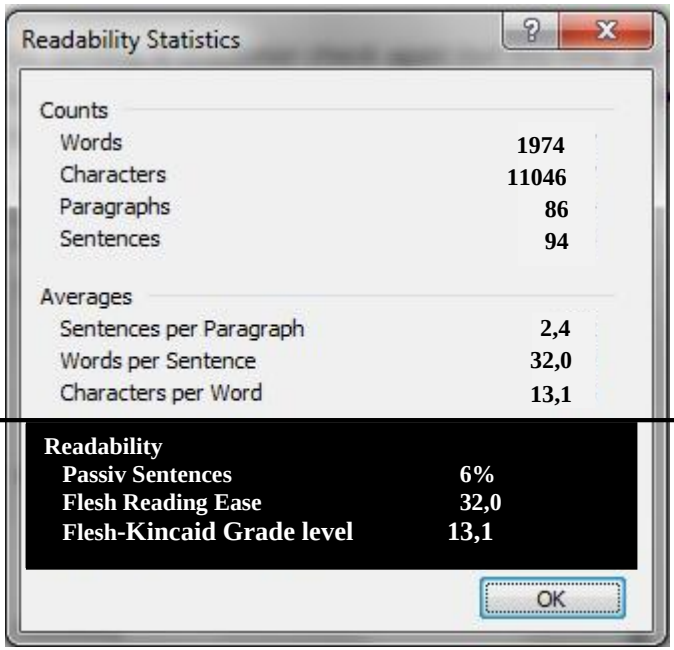
■დავაყენოთ ალამი***4.1*** ნახაზზე მოყვანილი ფანჯარა ***„Show readability statistics“***(«*Показывать статистику удобочитаемости*») ოფცი აში.

გადმოვიტანოთ *Word*-ში (ან *PDF*–ში) აკრეფილი ჩანაწერი. ამის შემდეგ დოკუმენტში თუ ავამუშავებთ მართლწერის შემოწმებას (ჩანართი ***Rever*** [*Рецензирование*], ღილაკი ***Proofing*** [*Правописание*]), მაშინ ორთოგრაფიის შემოწმების შემდეგ გაიხსნება ადვილწაკითხვადობის ***Readibility Statistics***(*Статистика удобочитаемости*) ფანჯარა (***4.2,ა*** ნახაზი), რომელშიც მოცემული მონაცემები დაფუძნებულია სიტყვებში მარცვლების, ხოლო წინადადებაში – სიტყვების საერთო რაოდენობაზე. რაც შეეხება ***ფლეშის ტესტის*** და ***ფლეშ-კინკეიდის სასკოლო ტესტის*** მეშვეობით გამოთვლილ ადვილწაკვთხვადობის მაჩვენებლებს, ისინი მოცემულ შემთხვევაში არ არის გამოტანილი.

განათლების სფეროში მომუშავე ამერიკელმა სპეციალისტმა ***რუდოლფ ფლეშმა*** (***Rudolf Flesch***, *1911–1986*) განსაზღვრა ტექსტის მახასიათებლები, რომლებიც აადვილებს ან აძნელებს მათ გაგებას. თავისი დასკვნები მან გამოიტანა იმ წასაკითხი ტექსტების ანალიზის საფუძველზე, რომლებსაც მასწავლებლები ტრადიციულად იყენებდნენ მოსწავლეების ცოდნის შესაფასებლად კლასიდან კლასში გადა-

ყვანის დროს. ***ფლეშმა*** დაადგინა, რომ მათზე გავლენას ახდენს ***100*** სიტყვაში არსებული მარცვლების რაოდენობა და წინადადების საშუალო სიგრძე. სწორედ ამ მახასიათებელთა თანაფარდობის განმსაზღ-

ა)



ბ)

1. Count – რაოდენობა; 2.**Vords –** სიტყვები; 3.**Characters** – სიმბოლოები; 4.**Paragraphs** – აბზაცები; 5.**Sentences** – სიტყვები; 6.**Averages** – საშუალო რაოდენობა; 7.**Sentences per Paragraph** – წინადადები აბზაცში; 8.**Word per Sentence -** სიტყვები წინადადებაში; 9. **Readibility –** ადვილწაკითხვადებადობა; 10. **Passive Sentences –** პასიური წინადადებები; 11. ***Flesh Realing Rase*** – წაკითხვის ფლეშისეული სიმარტივე; 12. Flesh-Kincaid Crade level ფლეშ-კინკეიდისეული დონე

***ნახ. 4.2.*(ა) -** *ადვილწაკითხვადობის სტატისტიკა*; **(ა)+ (ბ) –** *ინგლისურენოვანი ტექსტის ადვილადწაკითხვადობის სტატისტიკა.*

ვრელი ფორმულაა მჭიდროდ დაკავშირებული მოსწავლის მიერ ტექსტის გაგების დონესთან. ამ მეთოდიკამ მიიღო ***„ადვილსაკითხაობის ფლეშის ფორმულის სახელწოდება“***.მას აქვს ასეთი სახე:

$$K = 206{,}835 - 1{,}015 \times ASL - 84{,}6 \times ASW, \qquad (4.\ 1)$$

სადაც ***K*** არის ტექსტის სირთულის შეფასება, ***ASL*** – წინადადებაში სიტყვების საშუალო რაოდენობა, ხოლო ***ASW*** – წინადადებაში მარცვლების საერთო რაოდენობა.

100სიტყვიანი მონაკვეთების კვლევების შედეგად ფლეშმა ***1942*** წელს ტექსტის სირთულის შესაფასებლად შემოგვთავაზა ტესტების სირთულის შეფასების ***100***-ბალიანი სისტემა; ტექსტების შესაფასებლად გამოყენებული ბალების ზრდით მცირდება ტექსტის სირთულე. ***70-80***-ის ტოლ ბალს იღებს საკმაოდ ***ადვილად წასაკითხი ტექსტი***, ***60-65*** ბალს – საშუალო სირთულის ტექსტი, ხოლო ***30***-ზე ნაკლებ ბალს - ძნელად წასაკითხი ტექსტები. ***1948*** წელს ***ასოშეიტედ პრესის სააგენტომ*** შეამოწმა და მოიწონა ფლეშის მიერ შემოთავაზებული ტესტი. იგი ფართოდ გავრცელდა ***აშშ***-ის მთელ რიგ შტატებში საკანონმდებლო აქტების მიღების შემდეგ, რომლებიც მოითხოვდა დაზღვევის ხელშეკრულების ტექსტი საშუალო განათლების მქონე პირებისთვისაც ყოფილიყო გასაგები.

არსებობს***ფლეში-კინკეიდისსასკოლოტესტიც.*** იგი გამოიყენება აშშ-ის სკოლებში გამოყენებული ტექსტების სირთულის შესაფასებლად. მაგალითად, ***8,0*** ბალი ნიშნავს, რომ ეს დოკუმენტი შეიძლება გაიგოს ***მერვე კლასის მოსწავლემ***. სასურველია, რომ დოკუმენტების უმრავლესობა ფასდებოდეს ***7,0***-დან ***8,0*** ბალამდე. ***ფლეშ-კინკეიდის*** ტესტისათვის განკუთვნილ ფორმულას აქვს სახე:

$$K = 206{,}835 - 1{,}015 \times ASL - 84{,}6 \times ASW \qquad (4.\ 2)$$

დოკუმენტში გამოყენებული ენა იმაზეა დამოკიდებული, თუ როგორ მოწმდება მისი ადვილწაკითხვადობა ***MS Office***-ში. დოკუმენტი თუ შეიცავს რამდენიმე ენაზე დაწერილ ტექსტებს, ***Word***–ს შეუძლია ყველა მათგანის მართლწერა შეამოწმოს, მაგრამ რაც შეეხება ამ ტექსტების ადვილწაკითხვადობას, მას გამოაქვს მხოლოდ უკანასკნელი ტექსტის ადვილწაკითხვადობის მაჩვენებლები.

ინგლისურენოვანი ტექსტის ანალიზის დროს ადვილწაკითხვადობის სტატისტიკის ფანჯარაში ჩნდება როგორც ***ფლეშის***, ისე ***ფლეშ-კინკეიდის*** ტესტის მეშვეობით გამოთვლილი მაჩვენებლები. ***4.2,ა+ბ*** ნახაზზე ისინი შავ ფონზე თეთრი ასოებითაა ნაჩვენები. ამმონაცემების თანახმად შესამოწმებელი ტექსტი საკმაოდ რთულია, რადგან ***ფლეშის*** ტესტით მიღებული იქნა ***32*** ბალი, რაც ***60***-ზე მნიშვნელო-

ვნად ნაკლებია, ხოლო ***ფლეშ-კინკეიდის*** ტესტით მიღებული იქნა ***13,1*** ბალი, რაც ***8,0-ს*** მნიშვნელოვნად აღემატება.

კონკრეტული ***ენების სპეციფიკა*** გავლენას ახდენს (***4.1***) და (***4.2***) ფორმულებში არსებულ კოეფიციენტებზე. მათზე ***ქართული ენის*** სპეციფიკის გავლენის დასადგენად საჭირო სამუშაოები სამწუხაროდ არ ჩატარებულა. ***რუსულენოვანი ტექსტებისათვის*** ამ მიმართულებით გადადგმული იყო გარკვეული ნაბიჯები. ამ დროს გამოყენებული იქნა ***ოჟეგოვის*** რედაქციით გამოსული რუსული ენის ლექსიკონი (***39174*** სიტყვა) და ***მიულერის*** რედაქციით გამოსული ინგლისურ-რუსული ლექსიკონი (***41977*** სიტყვა). ჩატარებული კვლევების შედეგად (***4.1***) ფორმულამ მიიღო სახს:

$$K = 206{,}835 - 1{,}3 \times ASL - 60{,}1 \times ASW. \qquad (4.3)$$

აღნიშნული ფორმულის პირდაპირი გამოყენება შეუძლებელია, რადგან *Word* არ გვატყობინებს რუსულსიტყვებში მარცვლების რაოდენობას, ხოლო ინტერნეტში ვერ მოიძებნა სიტყვაში სიმბოლოების საშუალო რაოდენობაზე დაფუძნებული ფორმულა.

5. ტექსტის ოპტიკური ამოცნობა

სკანერი საშუალებას გვაძლევს კომპიუტერში გრაფიკული ინფორმაცია ქაღალდის ფურცლიდან შევიტანოთ. დღეისათვის არსებობს იმის საჭიროება, რომ ტექსტური ინფორმაცია ქაღალდიდან (წიგნიდან, ჟურნალიდან, გაზეთიდან) გადავიტანოთ კომპიუტერის მეხსიერებაში და ეს ინფორმაცია ტექსტურ ფაილებში შევინახოთ. ეს ძირითადად ***ელექტრონული ბიბლიოთეკების*** შესაქმნელადაა საჭირო, რომლებშიც თანამედროვე გამოცემების გარდა შეინახება დიდი ხნის წინ გამოსული გამოცემები, რომელთა ***ელექტრონული ვერსიები*** არ არსებობს. ***სკანირების*** შემდეგ ყველა, მათ შორის ტექსტური, ინფორმაცია სურათის სახეს იღებს. ასეთი ტექსტი შეიძლება დავათვალიეროთ და დავბეჭდოთ, ხოლო მისი ***რედაქტირება*** მხოლოდ ისეთი გრაფიკული რედაქტორითაა შესაძლებელი, რომელსაც აქვს ტექსტურ ინფორმაციასთან მუშაობისათვის საჭირო საშუალებები.

სურათის სახითწარმოდგენილი ინფორმაციის ტექსტურ დოკუმენტად გარდაქმნისათვის არსებობს ტექსტების ***ოპტიკური ამოცნობის*** სპეციალურიპროგრამები, რომელთა შორის ყველაზე უფრო ცნობილი და ხშირად გამოყენებადი პროგრამაა ***Fine Reader*** (ფირმა ***Abby***). ამოცნობის პროგრამების დახმარებით კომპიუტერი, ფიგურულურად რომ ვთქვათ, ნაბეჭდი და ხელნაწერი დოკუმენტების „კითხვას“ სწავლობს.

4.2. სპეციალურიტექსტებისფორმირებისსაკითხები

სახელმძღვანელოების, სამეცნიერო და ტექნიკური ტექსტების მომზადებისას გვიხდება მათში ჩავსვათ ფორმულები, სპეციფიკური აღნიშვნები და სხვადასხვა სახის სქემა. ასეთ რთულ შედგენილ ტექსტურ დოკუმენტებს ***სპეციალური ტექსტები*** ეწოდება.

1. ტექსტურ დოკუმენტში მათემატიკური ფორმულების ჩანერგვა

ტექსტურ დოკუმენტებში ხშირად სხვადასხვა მათემატიკური ფორმულის ჩასმაა საჭირო. ამ ამოცანის გადაწყვეტა ცდება ***ტექსტური რედაქტორის*** შესაძლებლობებს; მას მხოლოდ საბაზისო ფუნქციების შესრულება შეუძლია, რის შედეგადაც ზემოთაღნიშნული ***„plaintext“***-ი მიიღება. ***ტექსტურ პროცესორს*** ასეთი ტექსტების მომზადების გარდა სხვა მრავალი დამატებითი ფუნქციების შესრულებაც შეუძლია. ერთ-ერთი მათგანია ***„plaintext“***-თან მათემატიკური ობიექტების დაკავშირება მასში ამ ობიექტების ჩანერგვის მიზნით. ორი სხვადასხვა სახის ობიექტის დაკავშირებისა და ერთ-ერთ მათგანში მეორის ჩანერგვისათვის გამოიყენება ***OLE-ტექნოლოგია*** (იხ. გვ. ***102***).

OLE-ტექნოლოგიის რეალიზებისათვის ტექსტური პროცესორი შეწყვილებულია ***მათემატიკურ რედაქტორთან***. ეს უკანასკნელი ტექსტური პროცესორისათვის ქმნის ***„plain text“***-ში ჩასანერგ მათემატიკურ ფორმულებს, რომლებსაც ***OLE-ობიექტები*** ეწოდება. მაშასადამე, ***მათემატიკური რედაქტორის*** ფუნქციაა მოემსახუროს ოპერაციულ პროცესორს და მოამარაგოს იგი საჭირო ***OLE***-ობიექტებით, ამიტომ მას ***OLE-სერვერი*** უწოდებენ.

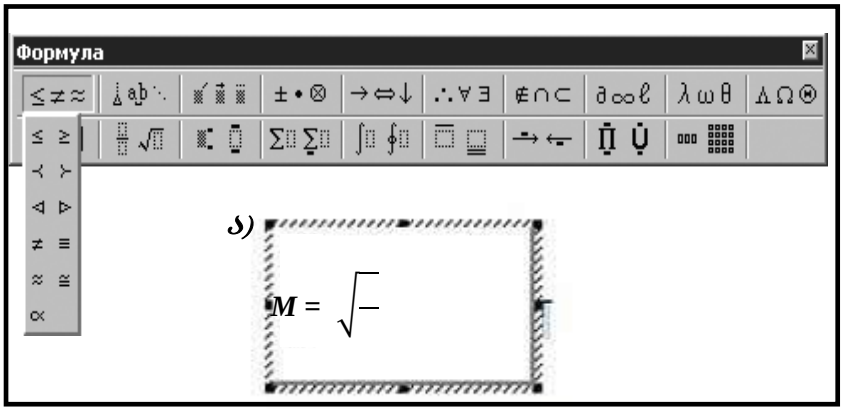
ტექსტურმა პროცესორი *OLE*-სერვერს შესასრულებლად უგზავნის გარკვეული სახის *OLE*-ობიექტებით მომარაგების მოთხოვნებს. სერვერისათვის მოთხოვნების გამგზავნ აპარატურულ ან პროგრამულ კომპონენტს ***კლიენტი*** ეწოდება. *OLE*-სერვერისაგან*OLE*-ობიექტების მომთხოვნ ცლიენტს, ბუნებრივია, ვუწოდოთ ***OLE-კლიენტი***.

OLE-ტექნოლოგიის რეალიზებისათვის ტექსტურ პროცესორად შევირჩიოთ ***Microsoft Word***-ი, ხოლო მათემატიკურ რედაქტორად – ***Microsoft Equation.*** ასეთი კონკრეტიზაცია არ არღვევს ზოგადობას,

რადგან ყოველივე ამ შემთხვევაში თქმული შეიძლება ადვილად გავავრცელოთ სხვა ტიპის ტექსტური პროცესორისა და მათემატიკური რედაქტორისაგან ფორმირებულ ტანდემზეც.

OLE-ტექნოლოგიის შერჩეული კომპონენტების შემთხვევაში *OLE*-კლიენტია *Microsoft Word, OLE*-სერვერი - *Microsoft Equation,* ხოლო *OLE*-ობიექტები - მათემატიკური ფორმულები.

*„**plain text**“*-ში *OLE*-ობიექტების ჩანერგვის შედეგად მიღება რთული ***შედგენილი ტექსტური დოკუმენტი***, რომელიც *OLE*-ობიექტების თავისებურ კონტეინერადაც შეგვიძლია განვიხილოთ.



ნახ.4.1. *ფორმულების რედაქტორ Microsoft Equation-ის ფანჯარა*

ტექსტური *Microsoft Word*პროცესორი(*OLE*-კლიენტი) შესაძლებლობას გვაძლევს *„plain text“*-ის სათანადო ადგილებზე ჩავნერგოთ (ჩავსვათ) მათემატიკური *Microsoft Equation*რედაქტორის(*OLE*-სერვერის) დახმარებით შექმნილი მათემატიკური ფორმულები (*OLE*-ობიექტები). სპეციალური ტექსტების შექმნის ასეთი ხერხი ერთ დოკუმენტში სხვადასხვა ობიექტების ინტეგრირების საშუალებას გვაძლევს. ამ დროს უნდა გვახსოვდეს, რომ ობიექტების რედაქტირება მაშინაა შესაძლებელი, თუ კომპიუტერზე დაყენებულია სათანადო ***პროგრამა-სერვერი***. მის ასამუშავებლად უნდა შევასრულოთ ბრძანება *Insert* ► *Equation*(*Вставка* ► *Формула*). პროგრამა-სერვერი ამოქმედდება დამონიტორის ეკრანზე გაიხსნება ***ფორმულების რედაქტორის ფანჯარა*** (ნახ. ***4.1***).

მათემატიკური ფორმულის ჩასანერგი არე მართკუთხოვანი ჩარჩოთია შემოხაზული (იხ. ნახ.***4.1.ა***), ხოლო ფორმულაში სიმბოლოების შესატანი ადგილები პუნქტირული კვადრატებითაა გამოსახული. ამ კვადრატებზე მაუსის კურსორის დაყენების შემთხვევაში შევძლებთ მათში შევიტანოთ სიმბოლოები კომპიუტერის კლავიატურიდან ან ახალი ფორმულები მათემატიკური პროცესორის ფანჯრიდან.

2. სპეციალური ტექსტების შემქმნელი პროგრამები

სპეციალური ტექსტების შესაქმნელად უფართოესი შესაძლებლობებითაა აღჭურვილი ამ ამოცანის გადასაწყვეტად სპეციალურად დამუშავებული პროგრამები. ასეთი პროგრამის მაგალითია ამერიკელი მათემატიკოსისა და დამპროგრამებლის, ფუნდამენტური მონოგრაფიის ***„The Art of Computer Programming“*** (*„Искусство программирования“*) ავტორის ***დონალდ კნუტის*** (***Donald Ervin Knuth, 1938- 2005***) მიერ დამუშავებული ***სამეცნიერო პუბლიკაციების T_EX სისტემა***. აღნიშნული მონოგრაფიის პირველი ტომი გამოვიდა ***1969*** წელს, როდესაც პოლიგრაფიაში ჯერ კიდევ ძველი ტექნოლოგიები გამოიყენებოდა. ეს ტექნოლოგიები წიგნის ხელახლა გამოცემის დროს მასში ცვლილებების შეტანას პრობლემატურს ხდიდა. საკუთარი წიგნის განახლებული ვარიანტის ოპერატიულად დამუშავებისათვის ***კნუტმა*** შექმნა ფორმულების, ნახაზებისა და ცხრილების შემცველი სამეცნიერო ტექსტების განახლებული ვარიანტის დაჩქარებული მომზადების ორიგინალური სისტემა.

კნუტმა აღნიშნული სისტემის დასათაურებისთვისაც ასევე ორიგინალური სახელი მოიფიქრა: ***T_EX*** მიღებულია ბერძნული სიტყვისაგან ***τέχνη***, რომელიც ქართულად ითარგმნება როგორც „ხელოვნება“, „ოსტატობა“. პროფესიონალების შეფასებით ***T_EX*** რთული მათემატიკური ფორმულების ასაკრეფად შექმნილი საუკეთესო სისტემაა. იგი დიდი პოპულარობით სარგებლობს აკადემიურ წრეებში, განსაკუთრებით კი მას მათემატიკოსები და ფიზიკოსები აფასებენ.

T_EX სისტემა დაპროგრამების სპეციალიზებულ ენაზეა დაფუძნებული. ენაც და მისთვის აუცილებელი ტრანსლატორიც ***კნუტის*** იერაა შექმნილი. ამ სისტემის გამოყენებისას ფორმულებიანი დოკუმენტი ***„მონიშვნების ენით“*** აღიწერება. ***T_EX*** სისტემისათვის განკუთვნილი ***საწყისი ფაილი*** წარმოადგენს სპეციალური სიმბოლოებიანი

და ბრძანებებიანი ტექსტის შემცველ დოკუმენტს. აღნიშნული სიმბოლოებისა და ბრძანებების დახმარებით სისტემას გადაეცემა ინფორმაცია სამეცნიერო ტექსტის ფორმატირების შესახებ. აღნიშნული ფაილი შეგვიძლია ნებისმიერი ტექსტური რედაქტორით შევქმნათ; ამ დროს აუცილებელია რომ მიღებული იქნეს ***დაფორმატების ყოველგვარი ელემენტებისაგან თავისუფალი ტექსტური ფაილი***, ე. ი. ***„plain text“***. ეს ნიშნავს, რომ ტექსტი არ უნდა შეიცავდეს არავითარ შრიფტულ ორიგინალობებს, გვერდებად დანაწილებებს და ა. შ. სპეციალური პროგრამა ახდენს ***tex*** გაფართოების მქონე ***საწყისი ფაილების*** ტრანსლირებას (გადათარგმნას) ***dvi*** (***DeVice Inderendent–*** *„მოწყობილობისაგან დამოუკიდებელი“*) გაფართოებებიან ***ფაილებად***, რომლებიც შემდგომში შე-იძლება ავსახოთ ეკრანზე ან დავბეჭდოთ.

არსებობს სტანდარტული TEX სისტემის გაფართოებები. მათი მაგალითებია $AMSTEX$,$LaTEX$ და $XymTEX$ პაკეტები. განვიხილოთ ისინი.

■ $AMSTEX$ ***პაკეტი*** განკუთვნილია მნიშვნელოვანი, მაგრამ ვიწრო წრის ამოცანების გადასაწყვეტად; მისი საშუალებით, კერძოდ, ხდება ამერიკული მათემატიკური საზოგადოების მიერ გამოცემული მათემატიკური ჟურნალებისათვის განკუთვნილი სტატიების, აგრეთვე ამავე საზოგადოების მიერ გამოცემული წიგნების დაკაბადონდება (იხ. გვ. ***139***).

■ $LaTEX$ ***პაკეტი*** საშუალებას გვაძლევს საწყის ფაილში მხოლოდ ერთი სიტყვის შეცვლით წიგნად გარდავქმნათ დიდი სტატია, მასში სარჩევი ერთი ბრძანებით ჩავსვათ, ხოლო განყოფილებების, თეორემებისა და ნახაზების დანომვრაზე თავი არ შევიწუხოთ, რადგან პროგრამა მათ თავად მიხედავს.

■ $XymTEX$ ***პაკეტი*** ქიმიური ფორმულების გასაფორმებლადაა განკუთვნილი.

შედარებით ვრცლად განვიხილოთ $LaTEX$ ***პაკეტი.*** მისი საწყისი $LaTEX$***-ფაილი*** უნდა იწყებოდეს დოკუმენტის სტილის განმსაზღვრელი ⊳documentclass ბრძანებით. მაგალითად, წიგნის სტილს განსაზღვრავს ⊳documentclass{book} ბრძანება. ფიგურულ ფრჩხილებში ჩასმული სიტყვა {book} გვიჩვენებს, რომ დოკუმენტი გაფორმდება წიგნის სახით. სტატიებისა და საქმიანი წერილების სახით დოკუმ-

ენტის გასაფორმებლად შესაბამისად გამოიყენება სტილი (კლასები) {articl} და {letter}. დოკუმენტის საწყისი ტექსტი არ უნდა შეიცავდეს გადატანებს (***LaTEX*** ამას თვითონ შეასრულებს). სიტყვები ერთმანეთისაგან პრობელებით, ხოლო აბზაცები – ცარიელი სტრიქონებით უნდა იყოს გაყოფილი. ფაილი end{document} ბრძანებით უნდა თავდებოდეს.

სტატიის საწყისი***LaTEX-ფაილს*** სტრუქტურული სახე ***4.2*** ნახაზზეა მოყვანილი.

```
\documentclass{article}
\begin{document}
აქ უნდა მოთავსდეს სტატიის ტექსტი
\end{document}
```

ნახ.4.2. LaTEX-ფაილს*სტრუქტურული სახე*

LaTEX-ში ფორმულები აიკრიფება სპეციალური ბრძანებებით. მაგალითად, ქვემოთ მოყვანილ სტრიქონს:

```
\frac{2}{\beta\sqrt{3\pi}}\exp\left(-\frac{(x-\delta)^3}
{2\beta^2}\right
```

შეესაბამება ფორმულა:

$$\frac{2}{\beta\sqrt{3\pi}}\exp\left(-\frac{(x-\delta)^3}{2\beta^2}\right)$$

LaTEX საშუალებას გვაძლევს ფორმულაში ჩავსვათ ფერადი სიმბოლოები. მაგალითად, თუ გვინდა რომ მიღებულ ფორმულაში ბერძნული ასო ***π წითლად იყოს დაბეჭდილი***, საჭიროა ზემოთ მოყვანილ ფორმულაში შევიტანოთ ცვლილება (ცვლილება მსხვილი შრიფტითაა დაბეჭდილი და ქვემოდანაცაა გახაზული):

```
\frac{2}{\beta\sqrt{3\color{Red}{\pi}}}\exp\left(-\frac{(x-\delta)^3}
{2\beta^2}\right
```

არსებობს ***LaTEX***-ს გაფრთოება, რომლის აღნიშვნაა ***LaTEX'***a. მისი უპირატესობაა ის, რომ იგი არ არის დამოკიდებული პლატფორმაზე (ოპერაციულ სისტემაზე) და, გარდა ამისა, იგი თავისუფლად (უფასოდ) ვრცელდება.

სამეცნიერო დოკუმენტაციის მოსამზადებელ სპეციალიზებულ პროგრამებს შორის უნდა გამოვყოთ ამერიკული ***MacKinchan Software***-ს მიერ ***2001*** წელს დამუშავებული ***ტექსტური რედაქტორი Scientific Word.*** იგი შეიძლება გამოვიყენოთ როგორც ავტონომიურად, ისე ***Scieentific WorkPlace*** პაკეტის შემადგენლობაში, რომელ-იც ტექსტური რედაქტორის გარდა შეიცავს ორ მათემატიკურ - ***Maple V*** და ***MuPAD*** - პროცესორს.

Scientific Word დოკუმენტებს ავტომატურად ***LaTEX*** ფორმატში შეინახავს. მისი დახმარებით სამეცნიერო ტექსტის მომზადების დროს ***მონიშვნების ენა*** საჭირო არ არის, რადგან ფორმულები აიკრიფება და ტექსტი ფორმატირდება ინსტრუმენტების პანელით.

Scientific Word განსაკუთრებით ფართოდ ისეთ სამეცნიერო გარემოში გამოიყენება, სადაც რთული სტრუქტურის ფორმულების აგების ხარისხი მაღალი უნდა იყოს და ტექსტები გაჯერებულია სამეცნიერო სიმბოლოებით.

4.3. საგამომცემლო სისტემები

საგამომცემლო სისტემების (*Publlishing Systems*) ძირითადი მიზანია ერთხელ შექმნილი ტექსტების ტირაჟირება, ე. ი. მრავალჯერადად გამოცემა. მისი მთავარი ამოცანებია: ტექსტებისა და მათი გრაფიკული მასალების შექმნა, მათთვის ტირაჟირებისათვის მოსახერხებელი ფორმის მიცემა და, ბოლოს, საკუთრივ ტირაჟირება.

სამაგიდო საგამომცემლო სისტემებად წოდებული კომპიუტერული პროგრამული საშუალები გვაწვდის გამოცემის დაბეჭდვამდე ჩასატარებელი საგამომცემლო სამუშაოს ავტომატიზებისათვის საჭირო საშუალებებს.

დაბეჭდვამდე მომზადება მომავალი გამოცემისათვის „გარეგნობის“ამსახავი ***მაკეტის*** შექმნით იწყება. ამ ეტაპზე ამოირჩევა: გვერდების ფორმატი, მათი ორიენტაციები და ველები, დიზაინის ელემენტები; დაისახება ტექსტის, ილუსტრაციებისა და სათაურების განთავსების ადგილები.

დამუშავებული მაკეტის საფუძველზე გვერდების შექმნას, ე. ი. გვერდებზე ტექსტის, გამოსახულებების, სათაურების, ლოგოტიპები-

სა (საფირმო სასაქონლო ნიშნებისა) და დიზაინის სხვა ელემენტების განთავსებას, ***დაკაბადონება*** ეწოდება.

ტექსტური და გრაფიკული მასალის გაერთიანებით მიღებულ მაკეტს, რომლის თითოეული გვერდი სრულიად ემთხვევა მომავალი გამოცემის შესაბამის გვერდებს, ***ორიგინალ-მაკეტი*** ეწოდება.

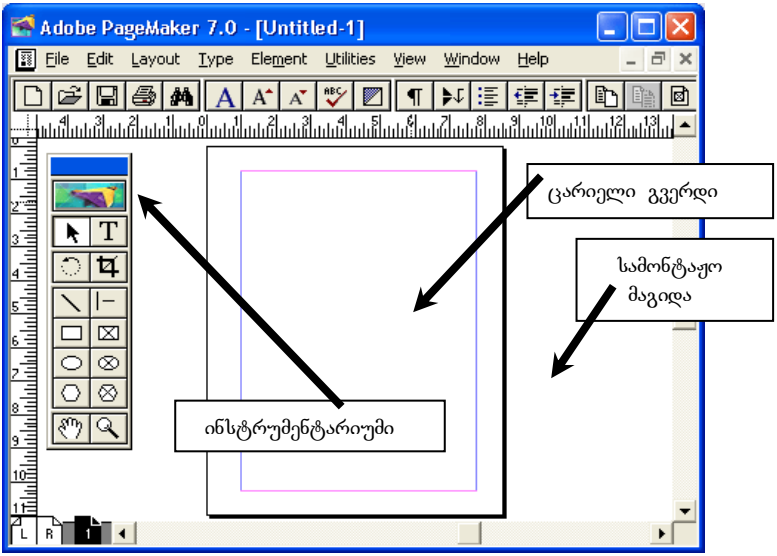
თანამედროვე ***ტექსტურ პროცესორებსა*** და ***საგამომცემლო სისტემებს*** შორის არსებობს მსგავსებებიცა და განსხვავებებიც. განვიხილოთ ორივე მათგანი.

■ ***მსგავსებები.*** ტექსტურ პროცესორებში რეალიზებულია საგამომცემლო სისტემებისათვის დამახასიათებელი ბევრი ფუნქცია: ტექსტის დაფორმატება, გვერდების მონიშვნა, ალფაბეტური და საგნობრივი მაჩვენებლებისა და სარჩევების შექმნა, მეხსიერების მართვა.

■ ***განსხვავებები.*** ტექსტურ პროცესორს აქვს მაკეტირების შეზღუდული შესაძლებლობები. იგი შეგვიძლია დაკაბადონებისათვის საჭირო ტექსტური მასალის მომზადების დროს მხოლოდ დამხმარე ინსტრუმენტად გამოვიყენოთ. საქმე ისაა, რომ საგამომცემლო სისტემა განკუთვნილია არა დიდი მოცულობის ტექსტის ასაკრეფად, არამედ ტექსტური და გრაფიკული ბლოკებისაგან რთული მაკეტის მიხედვით გამოცემის დაკაბადონებისათვის. ივარაუდება რომ ბლოკები (აკრეფილი ტექსტი, ფოტოსურათები, ნახატები) იქმნება ცალკეული პროგრამების (ტექსტური და გრაფიკული რედაქტორების) მეშვეობით, რომლებსაც ამისათვის საჭირო სპეციალური ფუნქციების შესრულება შეუძლია. ამიტომ მნიშვნელოვანია საგამომცემლო სისტემას ფაილური ფორმატების ფართო დიაპაზონის იმპორტირების უნარი ჰქონდეს. ***საგამომცემლო სისტემები*** შესაქმნელი დოკუმენტების ტიპოგრაფიულ აღწარმოებაზეა ორიენტირებული; ამიტომ მათში რეალიზებულია ისეთი ტიპოგრაფიული აწყობები, როგორიცაა ფერის კონტროლი, შრიფტის მახასიათებელთა ფაქიზი რეალიზატორები და ა. შ.

4.3 ნახაზზე მოყვანილია საგამომცემლო სისტემა *Adobe PageMaker*-ს ეკრანის სახე, რომელსაც იგი იღებს ბრძანება *File* ►*Nev* (Файл ►*Создать*)-ს შესრულების შემდეგ. მისი შიგთავსი პირობითად შეიძლება გავყოთ ***სამონტაჟო მაგიდად,*** რომელზეც ტარდება გამოსაქვეყნებელ მასალებთან ჩასატარებელი სამუშაოები და ინსრუმენტარიუმის შემცველი სისტემის ***ინტერფეისულ ნაწილად***. სამონტაჟო მაგიდის ცენტრში ახალი პუბლიკაციის ***ცარიელი გვერდია***

მოთავსებული. ***ორიგინალ-მაკეტის*** მომზადება გულისხმობს საჭირო მასალებით (ბლოკებით) ამ ცარიელი გვერდის შევსებას. ბლოკები შეგვიძლია გადავაადგილოთ, გავაძევოთ, ერთმანეთზე დავადოთ; ისეთი შთაბეჭდილება, თითქოს ბანქოს ბარათებივით ვშლით ტექსტებიან და სურათებიან ამონაჭრებს და მათ ვუმატებთ გასაფორმებელ გეომეტრიულ ელემენტებს – ხაზებს, ჩარჩოებს და ა. შ.



ნახ.4.3. *საგამომცემლო სისტემა PageMaker–ს ფანჯარა*

1990-იანი წლებიდან დაწყებული პოპულარობით სარგებლობდნენ საგამომცემლო სისტემები: *Adobe PageMaker, Adobe FrameMaker, Corel Ventura* და *QuarkXPress,*ხოლო დღეს გავრცელებულია *Adobe InDesign.*

V თავი
ცხრილური გამოთვლების ტექნოლოგიები

5.1. ელექტრონული ცხრილის სტრუქტურა და მონაცემთა ტიპები

ელექტრონული ცხრილი პროგრამაა, რომელიც ცხრილის მსგავსი ორგანზომილებიანი მასივების სახით წარმოდგენილ მონაცემებზე გამოთვლების ჩატარების საშუალებას გვაძლევს. მონაცემების „ფურცლებად“ ორგანიზების გზით ზოგიერთ პროგრამას მესამე განზომილება შემოაქვს.

ელექტრონული ცხრილი გამოთვლების ავტომატიზებისათვის საჭირო მოსახერხებელი ინსტრუმენტია. საბუღალტრო აღრიცხვის სფეროს ისეთი დოკუმენტები, როგორიცაა ბალანსები, საანგარიშო უწყისები, ხარჯთაღრიცხვები და ა. შ., ცხრილური ფორმით სრულდება. გარდა ამისა, მთელი რიგი ამოცანების რიცხვითი მეთოდებით გადაწყვეტაც ხრილური ფორმითაა მოსახერხებელი. ელექტრონულ ცხრილებში მათემატიკური ფორმულების გამოყენება გარკვეული რეალური სისტემის სხვადასხვა პარამეტრს შორის არსებული ურთიერთკავშირების წარმოდგენის საშუალებას გვაძლევს. მრავალი გამოთვლითი ამოცანის გადაწყვეტა, რომლებიც ადრე-დაპროგრამების-დახმარების გარეშე შეუძლებელი იყო, ელექტრონულ ცხრილებში მათემატიკური მოდელირების მეშვეობით გახდა შესაძლებელი.

ელექტრონული ცხრილების იდეა პირველად ავსტრიული წარმოშობის ამერიკელმა მეცნიერმა ***რიჰარდ მატეზიხმა*** (გერ. *Richard Mattesich*) ***1961*** წელს წამოაყენა სტატიაში *„Budgeting Models and System Simulation“.*

ელექტრონული ცხრილების, როგორც პროგრამული უზრუნველყოფის დამოუკიდებელი კლასის, საყოველთაოდ აღიარებული ფუძემდებელია ***დენიელ ბრიკლინი*** (*Daniel Bricklin, 1951წ.*), რომელმაც ***ბობი ფრენკსტონთან*** (*Bob Frankston, 1949 წ.*) ერთად ***1979*** წელს კომპიუტერ ***Apple II***-ისათვის დაამუშავა პროგრამა ***VisiCalc***. მისი წყალო-

ბით კომპიუტერი, რომელიც მანამდე ძირითადად ბავშვების სათამაშოდ გამოიყენებოდა, ბიზნესის ძირითად ინსტრუმენტად გარდაიქმნა.

შემდგომში ბაზარზე გამოჩნდა ამ კლასის უამრავი პროდუქტი, რომელთაგანაც უნდა გამოვყოთ *SuperCalc, Microsoft MultiPlan, Quaattro Pro, Lotus,* ***Microsoft Excel,*** *OpenOffice.org Calc,AppleWorks*. ელექტრონული ცხრილები მობილური ტელეფონებისა და ჯიბის პერსონალური კომპიუტერისათვისაც დამუშავდა; ასეთია, მაგალითად, *SpreadCE*.

ელექტრონული ცხრილი ***სტრიქონებისა*** და ***სვეტების*** მეშვეობით ***უჯრედებადაა*** დაყოფილი. სტრიქონები დანომრილია, ხოლო სვეტები აღნიშნულია ლათინური ალფაბეტის ასოებით. სვეტისა და სტრიქონის გადაკვეთაზე არსებულ უჯრედს მინიჭებული აქვს მისი წარმომქმნელი სვეტისა და სტრიქონის აღნიშვნების ერთმანეთზე მიწერით მიღებული სახელი (მაგალითად ***G3, N15*** და ა.შ.). ელექტრული ცხრილის იდენტიფიცირებისათვის ზოგჯერ გამოიყენება სახელების წარმოქმნის *RC* სისტემაც (*„**R**ov“ – სტრიქონი „**C**olumn“ - სვეტი*). ამ სისტემაში, მაგალითად, მე-17 სტრიქონისა და მე-8 სვეტის გადაკვეთაზე არსებულ უჯრედს ენიჭება *R17C8* სახელი.

მომხმარებელს ინფორმაცია ელექტრონული ცხრილის უჯრედებში შეაქვს. ინფორმაციას შეიძლება ჰქონდეს ტექსტის, რიცხვის ან ფორმულის სახე.

ტექსტი ცხრილის გასაფორმებლად (წანაწერების, სათაურების, ახსნა-განმარტებების გასაკეთებლად) გამოიყენება.

ელექტრონულ ცხრილში ***რიცხვები*** ჩვეულებრივი ან ექსპონენციალური ფორმით ჩაიწერება. ***ჩვეულებრივი ფორმის*** დროს რიცხვის მთელი ნაწილი წერტილით ან მძიმითაა გამოყოფილი წილადური ნაწილისაგან: 265.07; 37; 5.28. ***ექსპონენციალური ფორმით***, როგორც წესი, ძალიან დიდი ან ძალიან პატარა რიცხვები ჩაიწერება; მაგალითად, $1000000 = 1 \cdot 10^6$ და $0{,}00001 = 1 \cdot 10^{-5}$ რიცხვების ექსპონენციალური ფორმებია 1E6 და 1E-5. ექსპონენციალური ფორმით ჩაწერილი mEp რიცხვი (m ***მანტისად*** წოდებული მთელი ან წილადური რიცხვია, ხოლო p -ხარისხია) შეიძლება შემდეგნაირად წარმოვადგინოთ: $mEp = m \cdot 10^{p}$.

ელექტრონული ცხრილის ფორმულა ეწოდებაშეკრების, გამოკლების, გამრავლების, გაყოფისადაახარისხების ნიშნებით შეერ-

თებული რიცხვითი სიდიდეების, უჯრედთა მისამართების, ***ფუნქციების*** დასახელების შემცველ გამოსახულებას, რომლის წინ აუცილებლად დგას ***ტოლობის ნიშანი***. ფორმულა გამოთვლების რეალიზებისას ცხრილური პროცესორის მოქმედებას განსაზღვრავს. მისი ჩაწერის დროს მოქმედების შესრულების თანამიმდევრობა უნდა გავითვალიწინოთ. ფორმულაში ფრჩხილების არარსებობისას (ფრჩხილებში ჩასმული გამოთვლები პირველად სრულდება) მოქმედებები შემდეგი თანამიმდევრობით სრულდება: ***1)***ახარისხება (^); ***2)*** გამრავლება და გაყოფა (*და /); ***3)*** შეკრება და გამოკლება (+ და -).

ცხრილის უჯრედში ფორმულის ჩაწერისთანავე ცხრილური პროცესორი გამოითვლის ფორმულას და მიღებული შედეგი აისახება უჯრედში.

<u>მაგალითი 1.</u> ცნობილი გვერდების მიხედვით გამოვითვალოთ სამკუთხედის ფართობი. ჰერონის ფორმულის მიხედვით სამკუთხედის ფართობი გამოითვლება ფორმულით:

$$S = \sqrt{P \cdot (p - X) \cdot (p - Y) \cdot (p - Z)},$$

სადაც ***p*** არის სამკუთხედის ნახევარპერიმეტრი, ხოლო ***X, Y*** და Z – სამკუთხედის გვერდების სიგრძეები.

	A	B	C	D	E
1	გვერდების სიგრძეები				
2	X	Y	Z	ნახევარპერიმეტრი	ფართობი
3				= (A3+B3+C3)/2	= (D3*(D3-A3)*(D3-B3)*(D3-C3))^0,5

ნახ.. 5.1. *ელექტრონული ცხრილი ფორმულების ასახვის რეჟიმში*

ამოცანის გადასაწყვეტად საჭიროა ელექტრონული ცხრილი ***ფორმულების ასახვის რეჟიმში*** გადავიყვანოთ (ნახ. ***5. 1***).

სამკუთხედის ნახევარპერიმეტრისა და ფართობის გამოსათვლელად საწყისი მონაცემები სამკუთხედის გვერდებია, რომელთა მნიშვნელობები უნდა შევიტანოთ ***A3, B3*** და ***C3*** მისამართებიან უჯრედებში. ამ უჯრედებში არსებული რიცხვითი მნიშვნელობების გამოყენებით გამოითვლება სხვადასხვა სამკუთხედის ფართობები. ***2.5*** ნახაზზე ***მნიშვნელობების ასახვის რეჟიმში*** მყოფი ცხრილია მოყვანილი.

ამგვარად, ფორმულებისათვის განკუთვნილ საწყის მონაცემებზე მოქმედებების ჩასატარებლად ელექტრონულ ცხრილში ფუნქციონირებს ***გამოანგარიშების მექანიზმი.***

	A	B	C	D	E
1	გვერდების სიგრძეები				
2	X	Y	Z	ნახევარპერიმეტრი	ფართობი
3				7,50	9,92

ნახ. 5.2 *ელექტრონული ცხრილი მნიშვნელობების ასახვის რეჟიმში*

რამდენიმე სამკუთხედის ფართობების ერთბაშად გამოსათვლელად ცხრილში უნდა შევიტანოთ მათი გვერდების სიგრძეები და გადავაკოპიროთ ნახევარპერიმეტრისა და ფართობის გამოსათვლელი ფორმულები. ***5.3*** ნახაზზე მოცემულია ცხრილი ***ფორმულების ასახვის რეჟიმშია წარმოდგენილი***. მივაქციოთ ყურადღება იმ ფაქტს, რომ ფორმულების კოპირებისას იცვლება ფორმულაში შემავალ უჯრედთა მისამართები, ე. ი. მისამართები ფორმულების ადგილმდებარეობის ფუნქციაა. ამას ეწოდება ***ფარდობითი დამისამართების პრინციპი.***

	A	B	C	D	E
1	გვერდების სიგრძეები				
2	X	Y	Z	ნახევარპერიმეტრი	ფართობი
3				= (A3+B3+C3)/2	= (D3*(D3-A3)*(D3-B3)*(D3-C3))^0,5
4				= (A4+B4+C4)/2	= (D4*(D4-A4)*(D4-B4)*(D4-C4))^0,5
5				= (A5+B3+C3)/2	= (D5*(D5-A5)*(D5-B5)*(D5-C5))^0,5

ცხრ. 5.3 *შეფარდებითი დამისამართების მადემონსტრირებელი ცხრილი ფორმულების ასახვის რეჟიმში*

ელექტრონული ცხრილის ფარგლებში ფორმულის ყოველმა გადანაცვლებამ ფორმულაში რომ არ შეცვალოს უჯრედის მისამართი, ზოგჯერ გვჭირდება შეფარდებითი დამისამართების პრინციპის ნაცვლად გამოვიყენოთ ***აბსოლუტური ანუ უცვლელი მისამართი***. აბსოლუტური მისამართი **$** ნიშნის დახმარებით ფორმირდება. **$** ნიშნის ორჯერ გამოყენებისას (**$C$2**) მთლიანად მისამართი (სვეტი და სტრიქონი) ფიქსირდება. შეგვიძლია დავაფიქსიროთ მხოლოდ სვეტი (**$C2**) ან მხოლოდ სტრიქონი (**C$2**).

დავუშვათ, რომ სამკუთხედის **X**, **Y** და **Z**გვერდების სიგრძეები დეციმეტრებშია გამოსახული და გვინდა სამკუთხედის ფართობი გავზომოთ კვადრატულ სანტიმეტრებში, ე.ი მიღებული შედეგი ***10***-ზე გავამრავლოთ (***1*** დმ = ***10*** სმ). ამისათვის სამკუთხედის ფართობის გამოსათვლელი ფორმულის კოპირებისას საჭიროა **E1** უჯრედის მი-

	A	B	C	D	E
1	გვერდ. სიგრძეები			1 დმ =	10
2	X	Y	Z	ნახევარპერიმეტრი	ფართობი
3				= (A3+B3+C3)/2	= (D3*(D3-A3)*(D3-B3)*(D3-C3))^0,5*$E$1^2
4				= (A4+B4+C4)/2	= (D4*(D4-A4)*(D4-B4)*(D4-C4))^0,5*$E$1^2
5				= (A5+B5+C5)/2	= (D5*(D5-A5)*(D5-B5)*(D5-C5))^0,5*$ES1^2

ცხრ. 5.4. აბსოლუტური დამისამართების მადემონსტრირებელი ცხრილი ფორმულების ასახვის რეჟიმში

სამართი უცვლელი დარჩეს (ნახ. ***5.4***), რისთვისაც შეგვიძლია გამოვიყენოთ **$E$1** ან **E$1**მისამართი (უჯრედის მისამართში ფიქსირდება უჯრედის სტრიქონის ნომერი).

5.2. ჩაშენებული ფუნქციები. ფურცლებს შორის მონაცემების გადაცემა

Excel-ისმეშვეობით გამოთვლებისათვის სხვადასხვა სახის ფორმულა საჭირო. ორი რიცხვის შესაკრებად, მაგალითად, დაგვჭირდება ფორმულა =A1+A2. არგუმენტების რაოდენობის გაზრდისას იგი გართულდება და, მაგალითად, ***10*** რიცხვის შეკრების შემთხვევაში, მიიღებს სახეს:

=A1+A2+A3+A4+A5+A6+A7+A8+A9+A10. (***5.1***)

SUM () [СУММ ()]ფუნქცია, სადაც მრგვალ ფრჩხილებში არგუმენტებია მისათითებელი, (***5.1***) ფორმულის კომპაქტურად ჩაწერის საშუალებას გვაძლევს: =SUM(A1:A10); [= СУММ (A1:A10)].

ფორმულებში ფუნქციების გამოყენება ძალიან ხშირად არა მარტოფო რმულის გამარტივების, არამედ ისეთი გამოთვლების ჩატარების საშუალებასაც გვაძლევს, რომლებსაც ფორმულების გარეშე ვერ ჩავწერთ. დავუშვათ, რომ გვინდა A1:A10 დიაპაზონში არსებული რიცხვებიდან ვიპოვოთ მაქსიმალური რიცხვი. მოცემულ შემთხვევაში მონაცემების ნაკრებიდან მაქსიმალური მნიშვნელობის ამომრჩევი MAX და ფუნქციის გამოუყენებლად დასამული ამოცანის ჩაწერასაც ვერ შევძლებთ. ფორმულა =MAX(A1:A10) [=МАКС (A1:A10)] საშუალებას მოგვცემს გამოვითვალოთ A1:A10 დიაპაზონში არსებული მაქსიმალური მონაცემი.

ფუნქციები ცხრილის უჯრედებში არსებული მონაცემების რედაქტირების ან ცხრილის უჯრედებში მონაცემების პოვნის საშუალებასაც გვაძლევს. კერძოდ, ტექსტში არსებული ყველა სიტყვის პირველი ასოს ასომთავრულით შესაცვლელად უნდა გამოვიყენოთ PROPER [ПРОПНАЧ] ფუნქცია, ხოლო LOOKUP [ПРОСМОТР] ფუნქციის დახმარებით შეგვიძლია საჭირო მონაცემი ვიპოვოთ ერთ სტრიქონში, სვეტში ან მასივში. მასივის მარცხენა სვეტში ან ზედა სტრიქონში მონაცემის პოვნის საშუალებას შესაბამისად VLOOKUP [ВПР] და HLOOKUP [ГПР] ფუნქციები გვაძლევს.

ასეთი და მსგავსი ფუნქციები Excel-ის სტრუქტურაშია გათვალისწინებული და ამიტომ მათ ***ჩაშენებული ფუნქციები*** ეწოდება. ამგვარად, ***Excel-ში ჩაშენებული ფუნქცია*** წინასწარ განსაზღვრული ფორმულა ანფორმულების ერთობლიობაა. მას აქვს უნიკალური სახელი და ასე ჩაიწერება:

=ფუნქციისდასახელება (არგუმენტები),

სადაც ***ფუნქციის დასახელება*** არის ფუნქციის უნიკალური სახელი, ხოლო ***არგუმენტები*** – ფუნქციის შესასვლელი მონაცემები.

Excel-ის ჩაშენებული ფუნციები ***„შავი ყუთის“*** პრინციპით მოქმედებს. შესასვლელ მონაცემების (არგუმენტების) მიწოდებისას იგი გამოითვლის შედეგს, ან გადმოგვცემს გამოსასვლელ მონაცემებს. შედეგის ფორმირების პროცესი ჩვენთვის დაფარულია: ვერ დავინახავთ, რომელი ფორმულები გამოიყენება გამოთვლებისათვის. უჯრედში მხოლოდ შედეგი აისახება. მაგალითად,

=SUM (A1:A10;C1:C10)

ფუნქციის შესასავლელი მონაცემებია A1:A10 და C1:C10 დიაპაზონში არსებული რიცხვები, გამოსასვლელი კი ამ რიცხვების ჯამი.

ფუნქციას შეიძლება არ ჰქონდეს არგუმენტები, ე.ი. ცარიელი იყოს არგუმენტების სია. მაგალითად, =PI () ფუნქცია π-ს მნიშვნელობას, უსიტყვო შეთანხმებით, ***15*** ციფრიანი სიზუსტით მოგვაწვდის, ხოლო =TODAU() [=СЕГОДНЯ ()] ფუნქცია შეგვატყობინებს მიმდინარე თარიღს;

Excel-ში470-მდე ფუნქციაა ჩაშენებული. ისინი იყოფა შეთავსებადებოდის, *OLAR*ანალიზურ, მონაცემთა ბაზებთან სამუშაო„თარიღისა და დროის“, საინჟინრო, საფინანსო, საინფორმაციო, ლოგიკურ,

მიმთითებელ და ჩამსმელ, მათემატიკურ, სტატისტიკურ, ტექსტურ და ვებ ***კატეგორიებად***.მათი დაზეპირება შეუძლებელია და საჭირო არც არის:*Excel*-ში გათვალისწინებულია ჩაშენებული ფუნქციათა პოვნის სტანდარტული ხერხი. *Excel 2007*-ის შემთხვევაში ამისათვის ლენტზე არსებულ *Excel*-ის მთავარ მენიუდან *Formulas* (*Формулы*) პუნქტის ამორჩევის შემდეგ გამოჩნდება ზემოთ აღნიშნული კატეგორიები. არსებობს მეორე ხერხიც: კლავიატურაზე კლავიშების *Shift+F3* კომბინაციაზე თითის დაჭერითეკრანზე გამოვა ფანჯარა *„Insert Function“* (*„Мастер функций“*).

გავეცნოთ ზოგიერთ ჩაშენებულ ფუნქციას.

1. ჩაშენებული ლოგიკური ფუნქციები

ლოგიკური კატეგორიის ჩაშენებული ფუნქციებია პირობითი *IF* ფუნქცია, აგრეთვე *AND*[*И*],*OR*[*ИЛИ*], *NOT* [*НЕТ*], XOR [*ИСКЛИЛИ*] ფუნქციები.

პირობითი IF() ფუნქცია ელექტრონულ ცხრილში გამოიყენება გარკვეული პირობის შესამოწმებლად და მას აქვს შემდეგი სახე:

IF(< პირობა >;< გამოსახულება 1 >; < გამოსახულება2 >);

[ЕСЛИ(< Пирома >;< гамосахулеба1 >; < гамосахулеба2 >)]

< პირობა > მოცემულია ლოგიკური გამოსახულების სახით, რომელიც *TRUE* და *FALSE* [*ИСТИНА* და *ЛОЖЬ*] მნიშვნელობებიდან ერთ-ერთ მნიშვნელობას იღებს.

< გამოსახულება 1 > და < გამოსახულება 2 > შეიძლება იყოს რიცხვები, ფორმულები ან ტექსტები.

პირობითი ფუნქცია ჩაიწერება ელექტრონული ცხრილის უჯრედში და შემდეგნაირად გამოითვლება: თუ ***< პირობა >*** *ჭეშმარიტია, მაშინ მოცემული უჯრედის მნიშვნელობა ამოირჩევს* ***<გამოსახულება 1>****-ს, საწინააღმდეგო შემთხვევაში -* ***<გამოსახულება 2>****-ს.*

ლოგიკური გამოსახულებების ჩასაწერად გამოიყენება ***ურთიერთდამოკიდებულებათა ოპერაციები*** (<; >; <= (ნაკლებია ან ტოლია); >= (მეტია ან ტოლია); =; (არატოლია)) და ***ლოგიკური ოპერაციები***. ეს უკანასკნელები ელექტრონულ ცხრილებში *AND*-, *OR*-, *NOT*-, *XOR*-ფუნქციის სახითაა რეალიზებული. ლოგიკური ***AND-ფუნქცია*** იღებს *TRUE* [*ИСТИНА*] მნიშვნელობას თუ ყველა მისი არგუმენტს აქვს მნიშვნელობა *TRUE* [*ИСТИНА*]; ლოგიკური ***OR-ფუნქცია*** იღებს *TRUE* [*ИСТИНА*]მნიშვნელობას თუ მისი ერთი არგუმენტი მაინც

იღებს მნიშვნელობას *TRUE* [*ИСТИНА*]; ლოგიკური ***NOT-ფუნქცია*** არგუმენტის მნიშვნელობას ცვლის საწინააღმდეგო მნიშვნელობით; ***XOR-ფუნქცია*** იღებს მნიშვნელობას *TRUE* [*ИСТИНА*], თუ მისი არგუმენტები იღებს ურთიერთსაწინააღმდეგო მნიშვნელობებს.

მაგალითი 2. ჰერონის ფორმულით სამკუთხედის ფართობს ნებისმიერი საწყისი მონაცემებით ვერ გამოვითვლით, რადგან ნებისმიერი სამი რიცხვი არ შეიძლება იყოს სამკუთხედის გვერდების სიგრძე.გეომეტრიიდან ცნობილია, რომ სამკუთხედის ორი გვერდის სიგრძის ჯამი უნდა აღემატებოდეს მესამე გვერდის სიგრძეს; ამიტომ საკუთარი გვერდების***X, Y*** და ***Z***სიგრძეებით მოცემული სამკუთხედის ფართობის გამოთვლამდე ასეთი სამკუთხედის არსებობაში უნდა დავრწმუნდეთ.

ზემოთ ფორმულირებული პირობაშეიძლება შემდეგნაირად ჩავწეროთ:(*X+Y>Z AND Y+Z>X AND X+Z>Y*). ამ პირობის შესრულების შემთხვევაში შეიძლება ჰერონის ფორმულით გამოითვალოს სამკუთხედის ფართობი,საწინააღმდეგო შემთხვევაში კი ფორმულირებული უნდა იქნეს შეტყობინება:„სამკუთხედი არ არსებობს“.

	A	B	C	D	E
1	გვერდების სიგრძეები				
2	X	Y	Z	ნახევარპერიმეტრი	ფართობი
3				6	სამკუთხედი არ არსებობს
4				7,50	9,92

ნახ. 5.5. *ელექტრონული ცხრილი მნიშვნელობების ასახვის რეჟიმში, რომელიც ახდენს პირობითი ფუნქციის გამოყენების დემონსტრირებას*

5.5 ნახაზზე მოყვანილია ელექტრონული ცხრილის გამოსახულება, რომლის ***E3*** ცხრილში შეტანილია პირობითი ფუნქცია:

=IF(AND(A3+B3>C3;B3+C3>A3;A3+C3>B3);(D*(D3-A3)*(D3-B3)*(D3-C3))^0,5;„სამკუთხედი არ არსებობს“).

2. ჩაშენებული მათემატიკური ფუნქციები

ცხრილურ პროცესორ ***Excel2013***-ში გამოიყენება ***75*** ჩაშენებული მათემატიკური ფუნქცია. ასეთი ფუნქციებია SIN() [СИНУС], COS ()[КОСИНУС], SQRT() [КОРЕНЬ] და ა.შ. მრგვალ ფრჩხილებში მიეთითება არგუმენტი, რომლის აღსანიშნავად შეიძლება გამოვიყენოთ რიცხვითი კონსტანტა, ფორმულა, უჯრედის მისამართი ან

უჯრედების დიაპაზონი (ელექტრონული ცხრილის მართკუთხოვანი ფორმის არე).

სამკუთხედის ფართობის გამოსათვლელი ფორმულა:

(D3*(D3-A3)* (D3-B3)*(D3-C3)^0,5

შეგვიძლია ჩავწეროთ მათემატიკური ფუნქცია SQRT()-ის გამოყენებით: **= SQRT(D3*(D3-A3)* (D3-B3)*(D3-C3));**

[=КОРЕНЬ(D3*(D3-A3)* (D3-B3)*(D3-C3)].

3. ჩაშენებული სტატისტიკური ფუნქციები

სტატისტიკური კატეგორიის ფუნქციები საკმაოდ მრავალრიცხოვანია: მათი რაოდენობა ***104***-ს აღწევს. ყველაზე ხშირად გამოიყენება *AVERAGE* (СРЗНАЧА), MIN (МИН) დაMAX (МАКС) ფუნქციები. პირველი გამოითვლის არგუმენტების საშუალო მნიშვნელობას, ხოლო მე-ორე და მესამე შესაბამისად მინიმალური და მაქსიმალური სიდიდის არგუმენტებს.

მთელ რიგ ჩაშენებულ ფუნქციებში არგუმენტად გამოიყენება უჯრედების დიაპაზონი. იგი მოიცემა დიაპაზონის მარცხენა ზედა და მარჯვენა ქვედა კუთხეებში განთავსებულ უჯრედთა მისამართებით. განვიხილოთ სტატისტიკური ფუნქციების გამოყენების მაგალითი.

<u>მაგალითი3.</u>*5.1* ცხრილში მოყვანილია ***2007-2008*** წლებში მსოფლიოს ***9*** უმსხვილესი კორპორაციის სტატისტიკური მონაცემები. პასუხი გავცეთ შემდეგ კითხვებზე: ***ა.*** ჩამოთვლილი კორპორაციებიდან რამდენია განთავსებულ ***აშშ***-ში? ***ბ.*** რასუდრის იმ კომპანიების ჯამური ბრუნვა, რომელთა ბრუნვები ***100*** მილიარდ დოლლარს აღემატება? ***გ.*** რასუდრის ***აშშ***-ში განთავსებული კორპორაციების ჯამური ბრუნვა?

დასმული ამოცანა შეგვიძლია გადავწყვიტოთ ჩაშენებული სტატისტიკური COUNTIF() (СЧЕТЕСЛИ()) და მათემატიკური SUMIF() (СУММЕСЛИ()) ფუნქციების დახმარებით.

COUNTIF (СЧЕТЕСЛИ) ფუნქცია ასე ჩაიწერება:

= COUNTIF(დიაპაზონი; პირობა); [=СЧЕТЕСЛИ (დიაპაზონი; პირობა)].

SUMIF (СУММЕСЛИ) ფუნქცია ასე ჩაიწერება:

= SUMIF (დიაპაზონი; პირობა; ჯამვის დიაპაზონი);

[=СУММЕСЛИ(დიაპაზონი; პირობა; ჯამვის დიაპაზონი)].

დიაპაზონი თუ დაემთხვევა ჯამვის დიაპაზონს, მაშინ მხოლოდ ერთიდიაპაზონიმიეთითება. ამ შემთხვევაში ფუნქცია ჯამებს მოცემული პირობის დამაკმაყოფილებელ მნიშვნელობებს. SUMIF-ის ასეთი ფორმა გამოიყენება **ამოცანა 3**-ის „ბ“კითხვაზე პასუხის გასაცემად:

=SUMIF (D2:D9;“>100”);[=СУММЕСЛИ(D2:D9;“>100”)].

ამოცანა 3-ის „გ“ კითხვაზე პასუხის გასაცემად ორივე დიაპაზონის მითითებაა საჭირო, ე.ი. პირობა ვრცელდება უჯრედების B2:B9 დიაპაზონზე, ხოლო ჯამდება D2:D9 დიაპაზონის მონაცემები:

= SUMIF (B2:B9;“აშშ”; D2:D9);[=СУММЕСЛИ(B2:B9;“აშშ”;D2:D9)].

ცხრილი 5.1. *მონაცემები მსოფლიოს უმსხვილესი კორპორაციების შესახებ*

	A	B	C	D	E
1	**კომპანია**	**ქვეყანა**	**წარმოება**	**ბრუნვა (მილიარდი $)**	**თანამშრომლების რაოდ-ბა (ათასი)**
2	ექსონი	აშშ	ნავთოპროდუქტები	330	410
3	ტოიოტა მოტორსი	იაპონია	ავტომობილები	204	286
4	BMV	გერმანია	ავტომობილები	53	100
5	ჯენერალ მოტორსი	აშშ	ავტომობილები	148	252
6	IRI	იტალია	ლითონები	50	327
7	ტეკსაკო	აშშ	ნავთოპროდუქტები	66	72
8	ბრიტიშ პეტრო-ლეუმი	დიდი ბრიტანეთი	ნავთოპროდუქტები	295	102
9	ფორდ მოტორსი	აშშ	ავტომობილები	173	245

4. დავთრის სხვადასხვა ფურცელზე არსებული მონაცემების გამოყენება

ცხრილური *Microsoft Excel* პროცესორის გამოყენებისას მომხმარებელს საერთო ფაილში, ე. წ. ***დავთარში*** გაერთიანებულ რამდენიმე ელექტრონულ ცხრილთან, ანუ ***ფურცელთან*** აქვს საქმე. მას შეუძლია ცხრილური გამოთვლები ჩაატაროს ერთ ფურცელზე, ხოლო ფორმულებისათვის გამოიყენოს სხვადასხვა ფურცელზე განთავსებული მონაცემები.

განვიხილოთ დავთრის სხვადასხვა ფურცელზე განთავსებულ მონაცემებზე ცხრილური გამოთვლების ორგანიზების შემთხვევა.

კომუნალური მომსახურებისათვის საჭირო გაანგარიშებების ავტომატიზებისათვის შევქმნათ ელექტრონული ცხრილის ფაილი (***და-***

ვთარი), რომელსაც ექნება ორი ფურცელი – *„ტარიფები“* და *„ქვითარი“*. ფურცელ *„ტარიფებ“*-ზე (ნახ.*5.6,ა*) ჩავწეროთ ინფორმაცია კომუნალური მომსახურებაზე დაწესებული ტარიფების შესახებ, ხოლო ფურცელი *„ქვითარი“* (ნახ.*5.6,ბ*) გამოვიყენოთ მომსახურების გადასახდელად საჭირო ქვითრის მოსამზადებლად.

ა) ფურცელი „ტარიფები“

	A	B	C
1	**გადასახადის სახე**	**გაზომვის ერთეული**	**ტარიფი (ლარი)**
2	ელექტრომოხმარება	კვტ.ს	0,1454
3	წყალმომარაგება	სულთა რაოდენობა	3,8916
4	დასუფთავება	სულთა რაოდენობა	2

ბ) ფურცელი „ქვითარი“

	A	B
1	**მისამართი**	იასამნის ქუჩა 17-21
2	**მოხმარებული ენერგია**	120
3	**სულთა რაოდენობა**	5
4	**გადასახადის სახე**	**დარიცხულია**
5	ენერგომოხმარება	
6	წყალმომარაგება	
7	დასუფთავება	
8		
9	სულ მიმდინარე გადასახადი	

გ)

	A	B
1	**მისამართი**	იასამნის ქუჩა 17-21
2	**მოხმარებული ენერგია**	120
3	**სულთა რაოდენობა**	5
4	**გადასახადის სახე**	**დარიცხულია**
5	ენერგომოხმარება	=ტარიფები!C2*ქვითარი!$B$2
6	წყალმომარაგება	=ტარიფები!C3*ქვითარი!$B$3
7	დასუფთავება	=ტარიფები!C4*ქვითარი!$B$3
8		
9	სულ მიმდ. გადასახადი	=SUM(B5;B7)

ნახ.5.6. პირობითი დავთრის ფურცელი „ტარიფები“(ა),ფურცელი *„ქვითარი“ (ბ) და ფორმულებიანი ფურცელი „ქვითარი“ (გ)

სვეტ *„ტარიფები"*-ს შესავსებად საჭიროა გაზომვის ერთეული გადამრავლდეს შესაბამის ტარიფზე. სხვა ფურცლიდან საჭირო მონაცემების მისაღებად საჭიროა მივუთითოთ ფურცლის სახელი, შემდეგ–„!" ნიშანი და უჯრედის მისამართი. მაგალითად, უჯრედი ***ტარიფები! C2*** შეიცავს ტარიფფს ელექტრომოხმარებაზე. ფორმულების შეტანის შემდეგ ქვითარი იღებს ***5.6,3***-ნახაზზე მოყვანილ სახეს.

5.3. საქმიანი გრაფიკა

ნებისმიერ თანამედროვე ცხრილური პროცესორის ***საქმიანი გრაფიკის საშუალებებით*** შეგვიძლია ინფორმაცია წარმოვადგინოთ ***გრაფიკულად***, ე.ი. ***დიაგრამების*** სახით, რაც ამაღლებს ინფორმაციის აღქმის თვალსაჩინეობას. აღნიშნული საშუალებების გამოსაყენებლად საჭიროა:

1. გამოვყოთ (მოვნიშნოთ) ელექტრონული ცხრილის არე (უჯრედების დიაპაზონი ან რამდენიმე არამომიჯნავე დიაპაზონი);

2. ამოვირჩიოთ დიაგრამის ტიპი;

3. განვსაზღვროთ მონაცემების ამორჩევის თანამიმდევრობა (სვეტების თუ სტრიქონების მიხედვით).

სვეტების მიხედვით მონაცემების ამორჩევისას ***X***-კოორდინატები ამოირჩევა უჯრედების მონიშნული დიაპაზონის მარცხენა განაპირა სვეტიდან. დანარჩენი სვეტები შეიცავს დიაგრამის ***Y***-კოორდინატებს. ***სტრიქონების მიხედვით მონაცემების ამორჩევისას*** უჯრედების დიაპაზონის ყველაზე ზედა სტრიქონი წარმოადგენს ***X***-კოორდინატების სტრიქონს, დანარჩენი სტრიქონები შეიცავს დიაგრამის ***Y***-კოორდინატებს.

ცხრილური პროცესორების დახმარებით შეგვიძლია მრავალი ტიპის დიაგრამა ავაგოთ. ყველაზე ხშირად ვიყენებთ სვეტისებურ, წრფივ, წრიულ, იარუსისებურ და წერტილოვან დიაგრამებს. მოკლედ განვიხილოთ პირველი ორი მათგანი.

1. ჰისტოგრამა (სვეტისებური) დიაგრამა)

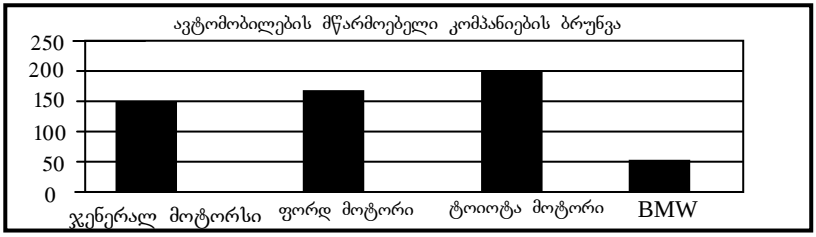
ჰისტოგრამა გამოიყენება რამდენიმე წერტილში ერთი ან რამდენიმე სიდიდის შესადარებლად. იგი შედგება ცალკეული სვეტებისაგან, რომელთა სიმაღლეები განისაზღვრება შესადარებელ სიდიდეთა მნიშვნელობებით. დიაგრამა თუ რამდენიმე სიდიდისათვის აიგ-

ება, მაშინ მათი იდენტიფიცირებისათვის გამოიყენება ***ლეგენდა*** – ამ სიდიდეთა პირობითი აღნიშვნები.

ცხრილი 5.2. *სვეტ„**წარმოება**“-ს მიხედვით დახარისხებული* ***5.2*** *ცხრილი*

	A	B	C	D	E
1	კომპანია	ქვეყანა	წარმოება	ბრუნვა (მილიარ-დი $)	თანამშრო-მელების რაოდ-ბა (ათასი)
2	ჯენერალ მოტორსი	აშშ	ავტომობილები	148	252
3	ფორდ მოტორსი	აშშ	ავტომობილები	173	245
4	ტოიოტა მოტორსი	იაპონია	ავტომობილები	204	286
5	BMV	გერმანია	ავტომობილები	53	100
6	IRI	იტალია	ლითონები	50	327
7	ბრიტიშ პეტრო-ლეუმი	დიდი ბრიტანეთი	ნავთოპროდუქტები	295	102
8	ტეკსაკო	აშშ	ნავთოპროდუქტები	66	72
9	ექსონი	აშშ	ნავთოპროდუქტები	330	410

მსოფლიოს უმსხვილესი კორპორაციების მონაცემების (იხ. ცხრ. ***5.1***) მიხედვით ავაგოთ ავტომობილების მწარმოებელი კომპანიების ბრუნვის ამსახავი ჰისტოგრამა. ამ შემთხვევაში სვეტების სიმაღლე გამოსახავს სვეტ ***„ბრუნვა (მილიარდი$)“***-ის რიცხვით მნიშვნელობებს. დიაგრამის აგების წინ უნდადავახარისხოთ (მოვაწესრიგოთ) სვეტ ***„წარმოება“***-ში არსებული მონაცემები (ცხრილი 5.2).



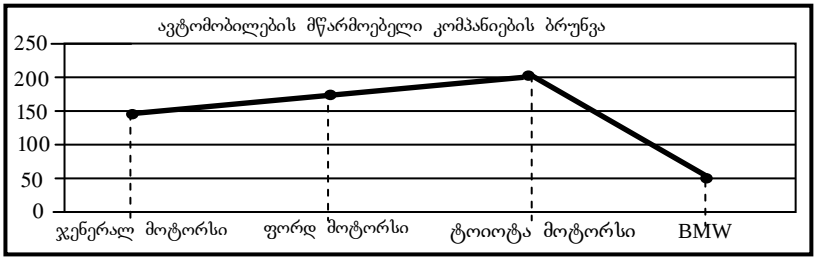
ნახ. 5.6. *ჰისტოგრამა*

მოვინიშნოთ უჯრედების **A2:A5; D2:D5** დიაპაზონი, როგორც ეს ნაჩვენებია ***5.2*** ცხრილში. მონაცემები შევირჩიოთ სვეტების მიხედ-

ვით. ჰისტოგრამა, რომლის ***X*** ღერძზე კორპორაციათა სახელებია აღნიშნული, ***5.6*** ნახაზზე არის მოყვანილი.

2. წრფივი დიაგრამა (გრაფიკი)

წრფივი დიაგრამა ანუ გრაფიკი იმისთვის გამოიყენება, რომ თვალი ვადევნოთ როგორ იცვლება ერთი ან რამდენიმე სიდიდე წერტილიდან წერტილამდე. მისი აგება ჰისტოგრამის აგების ანალოგურია, ოღონდ სიდიდეები სვეტების ნაცვლად წერტილებით გამოისახება და ეს წერტილები წრფეებითაა შეერთებული (ნახ.***5.7***).



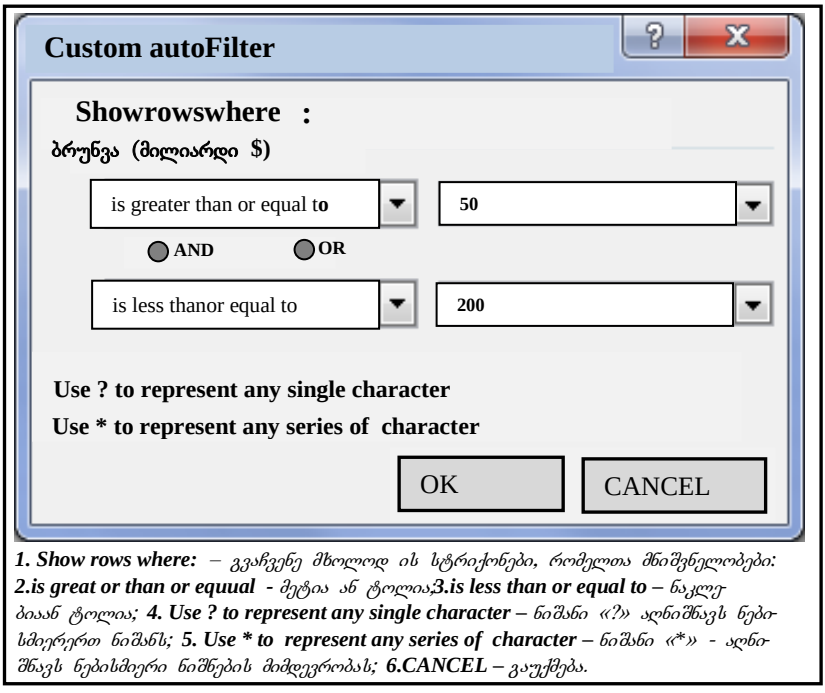
ნახ. 5.7. *წრფივი დიაგრამა (გრაფიკი)*

	A	B	C	D	E
1	კომპანია	ქვეყანა	წარმოება	ბრუნვა (მილიარდი $)	თანამშრ. რაოდ-ბა (ათასი)
2	ჯენერ	ზრდადობით დახარისხება	ავტომობილები	148	252
3	ფორდ	კლებადობით დახარისხება	ავტომობილები	173	245
4	ტოიო	(ყველა)	ავტომობილები	204	286
5	BMW	(პირველი 10 ...)	ავტომობილები	53	100
6	IRI	(პირობა)	ლითონები	50	327
7	ბრიტ ტრო	(დიდი ბრიტანეთი) (გერმანია) (იტალია)	ნავთოპროდუქტები	295	102
8	ტეკს	აშშ	ნავთოპროდუქტები	66	72
9	ექსო	(იაპონია)	ნავთოპროდუქტები	330	410

ნახ.5.8. ***Microsoft Excel***-ის ცხრილი

5.4. მონაცემების ფილტრაცია

თანამედროვე ცხრილურ პროცესორებში რეალიზებულია ***რელაციურ მონაცემთა ბაზების მართვის სისტემების*** ზოგიერთი, კერძოდ, ***მონაცემების ფილტრაციის***, შესაძლებლობა. ***მონაცემების ფილტრაციად*** ითვლება ამორჩევის პირობის დამაკმაყოფილებელი მონაცემების ამოღება ცხრილიდან. იგი მონაცემების კომპიუტერული ბაზიდან ინფორმაციის მოსაძებნად ***ფორმირებული მოთხოვნის*** ანალოგია. ცხრილურ ***MicrosoftExcel*** პროცესორში გამოყენებულია მონაცემების ფილტრაციის ორი ხერხი: ***ავტოფილტრაცია*** და ***გაფართოებული ფილტრაცია***. მათგან გავრცელებული და მოსახერხებელია ავტოფილტრაცია, რომელზც შევაჩერებთ თქვენს ყურადღებას. იგი საშუალებას გვაძლევს, ცხრილიდან ამოვირჩიოთ მარტივი პირობის დამაკმაყოფილებელი სტრიქონი.



1. Show rows where: *– გვაჩვენე მხოლოდ ის სტრიქონები, რომელთა მნიშვნელობები:* ***2.is great or than or equual*** *- მეტია ან ტოლია,****3.is less than or equal to*** *– ნაკლებიაან ტოლია;* ***4. Use ? to represent any single character*** *– ნიშანი «?» აღნიშნავს ნებისმიერთ ნიშანს;* ***5. Use * to represent any series of character*** *– ნიშანი «*» - აღნიშნავს ნებისმიერი ნიშნების მიმდევრობას;* ***6.CANCEL*** *– გაუქმება.*

ნახ. 5.9 *ფანჯარა „****სამომხმარებლო ავტოფილტრი****“*

მაგალითი 4. უმსხვილესი კორპორაციების შესახებ მონაცემების შემცველი ***5.1*** ცხრილიდან შევადგინოთ ***აშშ***-ის იმ კომპანიების სია, რომელთა ბრუნვა ***50***-დან ***200*** მილიარდ დოლარემდეა.

ამორჩევის პირობის შესაბამის ლოგიკური გამოსახულებაა:

(კომპანია=«აშშ»)AND(ბრუნვა(მლრდ$)>=50)AND ბრუნვა(მლრდ$)>=50).

[კომპანია=«აშშ»)И(ბრუნვა(მლრდ$)>=50)Иბრუნვა(მლრდ$)>=50]

გამოვყოთ ცხრილი და შევასრულოთ ბრძანება ***Data, Filter, AutoFilter*** (*Данные, Фильтр, Автофильр*). თავდაპირველად მივიღებთ აშშ-ში განთავსებული კომპანიების სიას. ამისათვის უნდა გავხსნათ სია, სვეტ „***ქვეყანა***“-ს შესაბამისი სია და ამოვირჩიოთ პუნქტი „**აშშ**“ (ნახ. ***5.8***). შემდეგ მიღებული ცხრილისათვის უნდა გამოვიყენოთ შერჩევის კიდევ ერთი პირობა. ამისათვის გავხსნათ მოცემული ცხრილის სვეტი „***ბრუნვა (მლრდ$)***“ (იხ.ნახ.***5.8***) და ამოვირჩიოთ პუნქტი ***პირობა***. გახსნილი ფანჯარა „***Custom AutoFilter***“ (*„Пользовательский автофильтр“*) ისე ავაწყოთ როგორც***5.9***ნ ახაზზეა ნაჩვენები.

ფილტრაციის შედეგად ვიღებთ ***5.10*** ნახაზზე წარმოდგენილ სამ სტრიქონს.

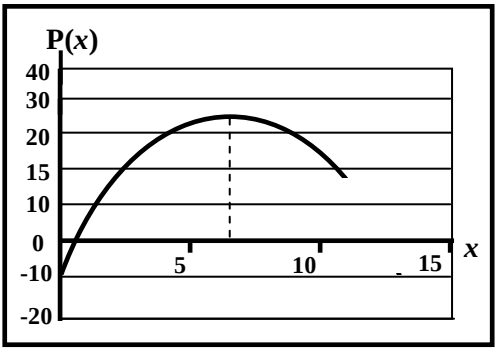
	A	B	C	D	E
	კომპანია	ქვეყანა	წარმოება	ბრუნვა (მილიარდი $)	თანამშრ. რაოდ-ბა (ათასი)
2	ჯენერალ მოტორსი	აშშ	ავტომობილები	148	252
3	ფორდ მოტორსი	აშშ	ავტომობილები	173	245
8	ტექსაკო	აშშ	ნავთობპროდუქტები	66	72

ნახ.5.10. *მაგალითი 4-ის გადაწყვეტა*

5.5. ოპტიმალური გადაწყვეტის პოვნის ამოცანა

ცხრილური პროცესორი ოპტიმალური გადაწყვეტის პოვნასთან დაკავშირებული ამოცანების გადაწყვეტის საშუალებას გვაძლევს. აღნიშნულის თვალსაჩინოდ ილუსტრირებისათვის განვიხილოთ შემდეგი პრაქტიკული სახის ამოცანა. განვსაზღვროთ გარკვეული საქონლის რა მოცულობის გაყიდვით მიიღებს წარმოება მაქსიმალურ მოგებას.

მოცემული ამოცანა მათემატიკურად ჩამოვაყალიბოთ. წარმოებული საქონლის რაოდენობა აღვნიშნოთ ***x*** ცვლადით, ხოლო ამ საქონლის საბაზრო ფასი – ***g*** ლარით.



ნახ. 5.11 P(x) *ფუნქციის გრაფიკი*

	A	B	C	D
1	a=	1	*x*	**P(*x*)**
2	b=	12		**=$B$4*C2-($B$1*C2^2+B2C*2+$B$3**
3	c=	10		
4	g=	25		

ნახ. 5.12. *ელექტრული ცხრილის საწყისი მდგომარეობა*

ცნობილია, რომ *x* რაოდენობის საქონლის დასამზადებლად საჭიროსაწრმოო დანახარჯები განისაზღვრება ფორმულით:

$$Z(x) = a \cdot x^3 + b \cdot x + c. \qquad (5.2)$$

x რაოდენობის პროდუქციის გაყიდვით საწარმოოს მიერ მიღებული *P(x)* მოგება უდრის საქონლის გაყიდვით მიღებულ შერმოსავალს მინუს პროდუქციის წარმოებაზე საწარმოოს მიერ გაწეული დანახარჯი:

$$P(x) = g \cdot x = Z(x). \qquad (5.3)$$

დავუშვათ, რომ მოცემულია ამ ამოცანის პარამეტრების მნიშვნელობები: $a = 1, b = 12, c = 10, g = 25$.

P(x) ფუნქციის გრაფიკიდან ჩანს (ნახ. ***5. 11***), რომ ამ ფუნქციას აქვს მაქსიმუმის წერტილი. აქედან გამომდინარე, საჭიროა ვიპოვოთ

ამ წერტილის შესაბამისი ***x*** აბსცისის მნიშვნელობა. ***P(x)*** ფუნქციას ***მიზნობრივი ფუნქცია*** ეწოდება.

ფორმულირებული მათემატიკური ამოცანი გადასაწყვეტად ცხრილურ *Microsoft Excel*პროცესორში გათვალისწინებულია *Excel Solver*-ად (Решающее устройство Excel) წოდებული ინსტრუმენტი. გამოთვლები სრულდება ***5. 12*** ნახაზზე ნაჩვენებ ელექტრონულ ცხრილში. მის ***D2*** უჯრედს ეწოდება ***მიზნობრივი უჯრედი***, ხოლო ***C2*** უჯრედს – ***ცვლებადი უჯრედი***; პირველში გამოითვლება მიზნობრივი ფუნქცია, ხოლო მეორეში - ***x***-ის მნიშვნელობა. გამოთვლების პროცესის განხილვამდე მოკლედ გავეცნოთ *Excel Solver* ინსტრუმენტს.

1. *Set Target Cell – მიზნობრივი უჯრედის დაყენება;* ***2.*** *Equal to – ტოლია;****3.*** *By Changing Cells – ცვლადი უჯრედი;* ***4.*** *Subjekt to the Constraints – შეზღუდვები;* ***5.*** *Solve- შესრულება;* ***6.*** *Close – დახურე;* ***7.*** *Add- დაუმატე;* ***8.*** *Change – შეცვალე;* ***9.*** *Resete all –ჩამოყარე.;* ***10.*** *Options პარამეტრები;* ***11.*** *Help –ცნობები;* ***12.****Change– შეცვალე;* ***13.*** *Delete - გააძევე*

ნახ. 5.13. *გადაწყვეტის მოძებნის Solver parameters ფანჯარა*

Excel Solver ბრძანებების სპეციალური ნაკრებია. იგი უმეტესწილად ბიზნესისა და საინჟინრო ამოცანების მოდელირებისა და ოპტიმიზებისთვისაა განკუთვნილი. იგი განსაკუთრებით სასარგებლოა წრფივი დაპროგრამებისა და ოპტიმიზაციის ამოცანების გადასაწყვეტად, ამიტომ მას ზოგჯერ ***„წრფივი დაპროგრამების ამომხსნელსაც“*** უწოდებენ.

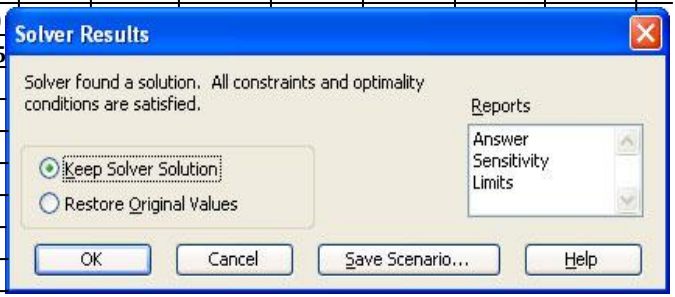
Excel 2003-დან დაწყებული *Solver*ზედნაშენი*Microsoft Excel*-ის ყველა ვერსიაშია გათვალისწინებული, მაგრამ უსიტყვო შეთანხმები ყველა მათგანში იგი ჩაყრთველია *Excel*-ისათვის მის დასამატებლად საჭიროა:

■ *Excel 2010, 2013, 2016*-შითითი დავაჭიროთ ჯერ «*file*»-ს და შემდეგ *Excel Options*»-ს (*Параметры*),ხოლო *Excel 2007*-ში – ჯერ *Mi-crosoft Offise*-ს და შემდეგ *„Excel Options“*-ს;

■ სადიალოგო ფანჯარა *„Excel Options“*-ის მარცხენა გვერდით პანელზე თითი დავაჭიროთ *Add-Ins*-ს (*надстройки*); გამოვა ფანჯარა *Excel Add-ins;* დავრწმუნდეთ, რომ ამ ფანჯრის «*Manage*» ველში *Excel Add-ins* ოპციაა არის ამორჩეულიდა ამის შემდეგთითი ღილაკ *Go*-ს (*Перейти*) დავაჭიროთ;

■ სადიალოგო ფანჯარა *Add-Ins*–ში დავაყენოთ ალამი *Solver Add-ins*და შემდეგ თითი *OK* ღილაკზედავაჭიროთ.

	A	B	C	D	E	F	G	H	I
1	a=	1	*x*	**P(*x*)**					
2	b=	12	*6,5*	**32,25**					
3	c=	10							
4	g=	25							
5									
6									
7									
8									
9									
10									
11									



1. Solverfound a solution. All constraints and optimality. Conditions are satisfied. – ამომხსნელმა იპოვა გადაწყვეტა. ყველა შეზღუდვა ოპტიმალურია. პირობები დაკმაყოფილებულია; **2.** Keep Solver Solution – ამონახსნი შეინახე ამომხსნელისათვის; **3.** Restore Original Values – აღადგინე საწყისი პირობები; **4.** Reports – შეტყობინება; **5.** Answer – შესაბამისობა; **5.** Sensitivity – აღქმადობა; **6.** Save Scenario – შეინახე სცენარი.

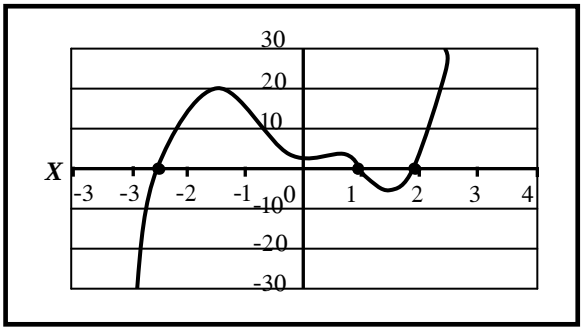
ნახ. 5. 14. ამოცანის გადაწყვეტის შედეგი

ახლა დავუბრუნდეთ ზემოთ აღნიშნული გამოთვლების პროცესს. გაღებულ *Solver Parameter* (*Поиск Решения*) ფანჯარაში (ნახ. ***5. 13***) მივუთითოთ, რომ ***C2*** უჯრედში (*წარმოების მოცულობა*) არსებული მნიშვნელობის ცვლილების გზით მოითხოვება მაქსიმალური გახდეს მიზნობრივ ***D2*** უჯრედში (*შემოსავალი*) არსებული მნიშვნელობა. ნა-

წარმოები საქონლისსაქონლის (***x***) მოცულობა არ შეიძლება უარყოფითი იყოს, ამიტომ შეზღუდვის *Subjekt to* შესრულების *Sohve* ღილაკზე თითის დაჭერის შემდეგ მივიღებთ ***5. 14*** ნახაზზე წარმოდგენილ შედეგს. ***6,5*** ერთეულის ტოლი წარმოების მოცულობის დროს მიიღება მაქსიმალური მოგება და იგი ***32,25*** ლარის ტოლი იქნება.

5.6. პარამეტრების შერჩევის ამოცანა

ცხრილური პროცესორი ოპტიმალური გადაწყვეტის ძიებასთან დაკავშირებული ზოგიერთი ამოცანის გადაწყვეტის საშუალებას გვაძლევს. დავუშვათ, რომ გვაინტერესებს გავიგოთ ***F(x)*** ფუნქცია ***x*** არგუმენტის რომელი მნიშვნელობისათვის იღებს ***a***-ს ტოლ მნიშვნელობას. ამის გასაგებად საჭიროა გადავწყვიტოთ ***F(x) = a*** სახის განტოლება. ***a***=0-ის შემთხვევაში ზემოთ მოყვანილი განტოლება იღებს ***F(x)*** =0 სახეს, ხოლო ***F(x)*** = $x^5 - 5x^3 + 2x^2 + 2$-ს შემთხვევაში იგი ასე ჩაიწერება:



ნახ. 5. 15. F(x) = $x^5 - 5x^3 + 2x^2 + 2$ *ფუნქციის გრაფიკი*

$$x^5 - 5x^3 + 2x^2 + 2 = 0 \qquad (5.4)$$

ზემოთ დასმული ამოცანის გადაწყვეტის საშუალებას გვაძლევს ცხრილურ *Microsoft Excel*პროცესორში არსებული ***Goal Seek*** [*Подбор параметра*]ინსტრუმენტი.

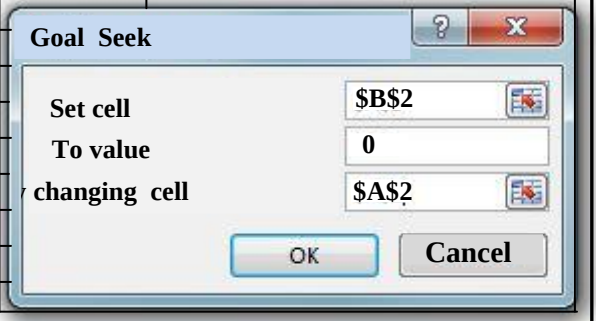
5.15 ნახაზზე მოყვანილია ***F(x)*** = $x^5 - 5x^3 + 2x^2 + 2$ ფუნქციის გრაფიკი. ამ გრაფიკის ***X*** ღერძის გადაკვეთის წერტილების აბსცისები არის (***5.2***) განტოლების ფესვები.

ა)

	A	B
1	*x*	*f(x)*
2	2	=A2^5-5*A2^3+2*A2^2+2
3		

ბ)

	A	B
1	*x*	*f(x)*
2	2	=A2^5-5*A2^3+2*A2^2+2
3		
4		
5		
6		
7		
8		
9		
10		
11		



გ)

	A	B
1	*x*	*f(x)*
2	*1,91629*	*0,00050*

1. Goal Seek [Подбор параметра**]** – პარამეტრის შერჩევა; **2. Set l [**Установить в ячейке**]** - აკრეფის უჯერდი; **3. Tovalue** [значение**]** - მნიშვნელობა; **4 . By changing cell [**Изменяя значение еячейки**]** - უჯრედის მნიშვნელობის ცვლილებისას.

ნახ. 5. 16. *განტოლების ფესვის პოვნის პროცესი*

გრაფიკიდან ჩანს, რომ $x^5 - 5x^3 + 2x^2 + 2 = 0$განტოლებას აქვს სამი ფესვი: $x \approx -2,5$, $x \approx 1$ და$x \approx 2$. აღნიშნული ფესვები ***0,1*** სიზუსტითაა განსაზღვრული. უფრო მაღალი სიზუსტის საჭიროებისამებრ შეგვიძლია წარმატებით გამოვიყენოთ ***Excel***-ის ელექტრონული ცხრილი. თავდაპირველად ვიპოვოთ $x \approx 2$ფესვის ზუსტი მნიშ-

ვნელობა. ამისათვის, უპირველეს ყოვლისა ეს მნიშვნელობა შევიტანოთ ელექტრონული ცხრილის ***A2***უჯრედში, ხოლო **=A2^5-5*A2^3+2*A2^2+2** გამოსახულება - ***B2***უჯრედში (ნახ. ***5.16,ა***); შემდეგ შევასრულოთ ბრძანება:

***Data*►*What-If Analysis* ► *Goal Seek*;**

[*Данные* ►*Анализ «что-если»* ► *Подбор параметра*].

გახსნილ *GoalSeek*-ის ფანჯარაში (ნახ. ***5. 16,ბ***) მივუთითებთ, რომ***A2*** უჯრედის შიგთავსის ცვლილებით ***B2*** (***F(x)*** = $x^5 - 5x^3 + 2x^2 + 2$)უჯრედშიმიღებული უნდა იქნეს***0***-ისტოლმნიშვნელო-ბა. გამოთვლით მიღებული იქნება, რომ ეს მოხდება მაშინ, როდესაც ***x ≈1,91629***(ნახ. ***5. 16,ბ***). ფესვის არც ეს მნიშვნელობაა აბსოლუტურად ზუსტი: ამ მნიშვნელობის შეტანისას (***5.4***) განტოლება უფრო მიუახლოვდება, მაგრამ არ გახდება ***0***-ის ტოლი. ჩვენ შეგვიძლია ვცვალოთ (ავამაღლოთ ან შევამციროთ) ფესვის განსაზღვრის სიზუსტე. სიზუსტეს სპეციალურად თუ არ დავსახავთ, მაშინ ***Goal Seek*** ინსტრუმენტი ფესვის მნიშვნელობას ***0,001***-ის ტოლი სიზუსტით გამოითვლის.

განტოლების დანარჩენი ფესვების ზუსტი მნიშვნელობების საპოვნელად საჭიროა ელექტრონულ ცხრილის ***A2***უჯრაში მიმდევრობით შევიტანოთ მათი ***x≈ 1*** და ***x≈ -2,5*** მნიშვნელობები და გავიმეოროთ ზემოთ აღწერილი ოპერაციები.

ლიტერატურა

1. ***დუნდუა ა.*** კომპიუტერული სისტემებისა და საინფორმაციო ტექნოლოგიების თეორიული საფუძვლები – თბ. „ტექნიკური უნივერსიტეტი“, ***2014.*** – 258 გვ.

2. ***დუნდუა ა.*** კომპიუტერული სისტემების ტექნიკური საშუალებები (***HARDVARE***) – თბ. „ტექნიკური უნივერსიტეტი“, ***2018.*** – 143 გვ.

3. ***მაჭარაშვილი გ.*** ოპერაციულ სისტემათა საფუძვლები. - თბ.„ტექნიკური უნივერსიტეტი“ , ***2009.*** – 107 გვ.

4. ***მაჭარაძე თ., წვერაიძე ზ.*** ინფორმატიკის საფუძგვლები. - თბ.:„ტექნიკური უნივერსიტეტი“, ***2009.*** – 342 გვ.

5. ***მაჭარაძე თ., წვერაიძე ზ.*** კომპიუტერები და კომპიუტერული ტექნოლოგიები. - თბ.„ტექნიკურიუნივერსიტეტი“ , ***2009.*** – 363 გვ.

6. Информатика / под ред. Трофимова В.В. – М. Издательство Юрайт, ***2013***. – 917 с.

7. ***Иртегов Д.В.*** Введение в операционные системы. – СПБ. «БХВ-Петербург», ***2014.*** – 1008 с.

8. ***Ляхович В.Ф., Молодцов В.А., Рыжкова Н.Б.*** Основы информатики. – М. КНОРУС, ***2016*** – 348 с.

9. ***Новожилов О.П.*** Информфтика.– М. Издательство Юрайт, ***2012***. – 564 с.

10. ***Олифер В., Олифер Н.*** Сетевые операционные системы. – М. Питер, ***2009***. – 672 с.

11. ***Операционные системы /*** Под ред. Спиридонова Э.С. – М.Книжный дом „ЛИБРОКОМ, ***2014***. – 352 с.

12. ***Таненбаум. Современные операционные системы.*** - СПБ. „БХВ-Петербург“, ***2014.*** – 1120 с.

13. ***Хлебников А.А.*** Информационные технологии. - М. КНОРУС, ***2014***. – 472 с.

15. http://www.dialektika.com/PDF/978-5-8459-1669-3/part.pdf

16. http://karpov-k.me/computernaya-nauka/os/344-failovaya-sistema-fizichiski

17. https://studopedia.su/1_5766_logicheskaya-i-fizicheskaya-organizatsiya-fayla.html.

18. http://book.kbsu.ru/theory/chapter3/unix.html.

19. https://xreferat.com/33/4928-1-pol-zovatel-skiiy-interfeiys.html.

რედაქტორი ბ. ცხადაძე

კომპიუტერული უზრუნველყოფა ალექსანდრე დუნდუასი

გადაეცა წარმოებას 13.09.2019. ხელმოწერილია დასაბეჭდად 03.10.2019. ქაღალდის ზომა 60X84 1/16. პირობითი ნაბეჭდი თაბახი 10. №3157.

საგამომცემლო სახლი ,,ტექნიკური უნივერსიტეტი‘‘,
თბილისი, კოსტავას 77

Verba volant,
scripta manent