9.1 Негізгі және фондық жіп жөніндегі түсінік

Бастапқыда жіптер негізгілер ретінде құрастырылады, бұл қосымша осындай жіптердің бірі орындалғанша дейін аяқталмайтынын білдіреді. C# тілінде фондық жіпдер деген түсінік бар, олар барлық негізгі жіптер біткен соң сол мезгілде аяқталады (олар қосымшаның өмірін «ұзартпайды» деп саналады).

Жіп мәртебесі - негізгі немесе фондық, логикалық типі бар IsBackground қасиетінің күйімен анықталады.

Оқу бағдарламасы ретінде кідіртудің әртүрлі уақытымен екі жіптің жұмысын қарастырайық:

Бірінші жағдайда `nt1.IsBackground = true;` болғанда – бірінші жіп фондық, бағдарлама екінші жіп жұмысының соңында өз жұмысын аяқтайды. Екінші жағдайда (екінші жіп фондық болады) бағдарлама өз жұмысын екінші жіп жұмысын аяқтағанша дейін аяқтайды. «Фондық ағын осындай тәсілмен аяқталғанда, ағын ішіндегі барлық **finally** блоктары ескерілмейді. **Finally**-да кодтың орындалмауы әдеттегінде қолайсыз болғандықтан, қажетті таймаутқа басып (**Thread.Join** көмегімен), бағдарламадан шығар алдында барлық фондық ағындардың аяқталуын тоқсан дұрыс болады. Егерде қандай да бір себептер бойынша жұмыс ағыны берілген уақытта аяқталмаса, оны апаттық түрде аяқтап көруге болады (**Thread.Abort**).

Жұмыс ағынның фондық ағынға ауысуы қосымшаны аяқтаудың соңғы мүмкіндігі болады, өйткені сөнуші /***умирающий***/ негізгі ағын қосымшаға аяқталуға мүмкіндік бермейді. Тұрып қалған негізгі ағын WindowsForms қосымшаларында әсіресе қауіпті, өйткені қосымша оның басты ағыны аяқталғанда аяқталады (пайдаланушы үшін), бірақ үдеріс жалғаса береді. Міндеттер диспетчерінде оның орындалып жатқан файлының атауы орындалып жатқан үдерістердің тізімінде қалсада, ол қосымшалар тізімінен жоғалып кетеді. Пайдаланушы оны тауып тоқтатқанша /***прибьет***/ дейін, үдеріс ресурстардың тұтынуын жалғастырады және де қосымшаның қайтадан қосылған данасының қосылуына немесе қалыпты қызмет етуіне кедергі жасайды.»

9.2 Жіптің басымдылығы жөніндегі түсінік

Жіптің сипаттамаларын анықтағанда, жіп басымдылығын Priority қасиетінің көмегімен беруге болады. C# тілінде жіпке беруге болатын Lowest, BelowNormal, Normal, AboveNormal, Highest деген басымдылықтың 5 градациясы бар. Жіптің басымдылығы жіпке басқа жіптерге қарағанда қаншама уақыт берілетінімен анықталады. Жіптің басымдылықтарын зерттеу үшін екі негізгі жіптермен, бірақ әртүрлі басымдылықтармен алдыңғы бағдарламаны пайдаланамыз.

9.3 Үдерісті анықтау

Windows-та орындалатын қосымшада пайдаланылатын жүйелік ресурстар жататын ядроның объектісі үдеріс деп аталады. Әдеттегінде, орындалып жатқан қосымшаны үдеріс деп атайды.

Әр үдерістің орындалуы басты жіптің орындалуынан басталады – консолдық қосымшалар үшін ол Main әдісі болуда.

Үдерісті орындау уақытында үдерістің басқа жіптері де немесе мүлдем басқа үдерістер де құрастырылу мүмкін.

Үдеріс барлық жіптердің жұмысы біткенде аяқталады.

Windows-та әр үдеріс келесі жүйелік ресурстарға ие болады:

- виртуалды мекенжайлық кеңестікке;

-қауіпсіздік жүйесінің ақпаратымен қатынау маркерімен;

- ядро объектілерінің дескрипторларын сақтау үшін кестемен.

Жіппен ұқсас әр үдеріс қызметтік бағдарламалармен пайдаланылатын өз идентификаторына ие.

9.4 Жұмыс істеп тұрған қосымшадан үдерісті құрастыру

Жұмыс қосымшадан қосымшаны қосу үшін (үдерісті құрастыру) Process сыныбын пайдалану қажет.

Process сыныбында осы дәрістің қосымшасында ұсынылған қасиеттер мен әдістердің жиынтығы бар. Тізім `Process` сөзін анықтағанда және F1 батырмасын басқанда орта анықтамасының көмегімен алынған.

Қосымшаға үдерістің табысты қосу үшін System.Diagnostics атаулар кеңестігін қосу қажет.

Process сыныбы Process.Start(FileName) немесе Process.Start() әдісін шақырту жолымен үдерістерді қосу үшін пайдаланылу мүмкін.

Process.Start(FileName) әдісін қосу алдында әдіс параметрі ретінде қосу үшін үдеріс файлының атауын (FileName қасиеттері) немесе оған деген толық жолды беру қажет.

Жаңа үдеріс объектісін құруға, FileName қаисетіне мән беруге және параметрлерсіз Process.Start() әдісінің көмегімен үдерісті қосуға болады. Бағдарламаның жұмысы «ПасьянсКосынка» ойынды қосуда болуда. Ойын соңында «Enter» пернесін басып «Блокнот» редакторын қосу керек. Блокнотпен жұмыс аяқталған соң «Enter» пернесіне басумен консолдық терезені жабу керек.

Process сыныбының әдістерін пайдалануға болады, мысалы, GetProcesses жұмыс істеп тұрған үдерістің немесе үдерістердің сипаттамаларын зерделеу үшін.

Келесі оқу бағдарламасында ағымдағы мезетке жұмыс істеп отырған адамның компьютерінде алғашқы үдерістің кейбір сипаттамалары қарастырылады.

10.1 Әрекет жөніндегі түсінік

Жіптің контекстін өзгертетін пәрмендердің кез келген реттілігі әрекет деп аталады. Жіп контекстінің анықтауы 2 дәрісте қарастырылады – бұл жіп өзінің орындалуы кезінде жүгіне алатын компьютер жадысының саласы.

Әрекет өзінің орындалуы кезінде үзілмесе, және әрекеттің өзімен ғана өзгертілсе, үздіксіз деп аталады.

Әрекет тек микропроцессордың үзу сигналымен үзілу мүмкін.

Алғашқы талап тек бірпроцессорлық жүйелерге ғана әділді, оларда әрекет орындау уақытына үзуге де тыйым салуға болады.

Мультипроцессорлық жүйелері үшін әрекеттер қосарланып әртүрлі жіптермен орындалу және бір-бірімен қиылысу мүмкін. Сондықтанда мультипроцессорлық жүйелер үшін әрекеттердің тоқтаусыздығы бір процессорда

орындалатын әрекетке басқа процессорда орындалып жатқан әрекетті өзгертуге тыйым салумен қамтамасыз етіледі.

Сонымен, мультипроцессорлық жүйелерде қосарланған жіптер жұмысының келісушілігі әрекеттердің тоқтаусыздығын қамтамасыз етудің маңызды міндеті.

Осы міндетті шешудің бір нұсқаларының бірі жіптер жұмысының синхрондалуы болуда.

Жіптер өздерінің арасында алмастыратын сигналдардың көмегімен оларды орындаудың қандай да бір бекітілген реттілігіне қол жеткізу жіптердің синхрондалуы деп аталады.

Синхрондалу болжанатын нәтижені алу үшін жіптердің әрекеттерін үйлестіру [Албахари б.739.]

10.2 Жіптердің синхрондау қасиеттерінің жіктелуі

Осы дәрісте бір үдерісте жіптердің синхрондалу мысаларындағы синхрондалудың теориялық аспектілерін қарастырайық.

Әдеттегінде, бағдарламаның негізгі әдісі (Main) басты жіп деп аталады, ал басты жіп үшін кебір міндеттерді шешетін әдістер қосымша немесе екінші жіптер деп аталады.

Әрине, осы жіптердің жұмысы реттелуге тиіс. Жіптер жұмысының реттелгендігінен басқа айқын (оқулық емес) бағдарламалар кейбір деректердің бірлескен пайдалану міндеттерін шешуге тиіс (жіптерге деректерді және жіптер арасында деректерді тапсырудың жеке сұрақтарын біз алдыңғы дәрістерде қарастырдық), бірақ синхрондау теорияларын зерделеу үшін осы міндеттерді жеке-жеке қорастырған тиімді.

«Синхрондаудың барлық қасиеттерін шартты түрде төр санатқа бөлуге болады:

- жаһандық айнымалылар және жіптің орыдалуын тоқтатудың қарапайым әдістері жататын жіптердің синхрондалуының қарапайым қасиеттері (Sleep иJoin);
- синхрондау объектілері (Mutexи Semaphore) және lock операторы жататын арнайы блоктаушы конструкциялар;
- арнайы сигналдарды беруге және қабылдап алуға негізделген конструкциялар. Осы конструкциялар басқа жіптен сигнал алғанша дейін жіптің орындалуын тоқтатып тұрады. Әдеттегінде, сигналдық механизмді пайдаланатын екі осындай конструкциялар қолданылады, олар оқиғаларды күту дескрипторлары және Monitor сыныбының Wait/Pulse әдістері;
- Interlocked сыныбын жатқызатын жіптердің орындалуын талап етпейтін құралдар (синхрондау қасиеттерін бөгет етпейтін), арнайы негізгі сөз volatile, ол оқу – деректерді жазу операцияларының кэштеуіне тыйым салуға мүмкіндік береді.»[Албахари б. 739-740].

Жіптердің синхрондауының аталған қасиеттерінен басқа, ContextBoundObject сыныбымен және Synchronization атрибутымен жүзеге асырылатын жіптің ресурстарына қатынаудың автоматты синхрондау (блоктау) нұсқасы бар. Оларды біз бесінші санат құралдары деп атаймыз.

Синхрондаудың қандай да бір қасиетін таңдау шешілетін міндетпен анықталады және бағдарламашыға байланысты.

Синхронда құрадларының зертелуін ең қарапайым санаттардан бастаймыз.

Албахари Интернетте жарияланған мақаларының бірінде жіптер синхрондалу қасиеттерінің келесі тізімін келтіреді (мақала аудармашысы оларды ағындар деп атйды):

10.3 Жіптер синхрондалуының қарапайым қасиеттері

Жіптер синхрондалуының теорияларын қарастыру кезінде, дәстүрлі түрде олардың жұмысының кезектілігін көзбен бақылау үшін кейбір әртүрлі мәндердің циклдық қорытындысы қолданылады, мысалы, әртүрлі символдар немесе сандар. Дәстүрден алшақтамайық, біздің жіптердің жұмысы да монитор экранның консолдық терезесіне сандық мәндерді шығаруда болады (нақты жіптер басты жіп үшін жұмыс атқарады).

Басты жіптен басқа, біздің бағдарламада қосымша екі жіп бар деп болжайық. Басты жіп консолдық терезеге 0-ден 9-ға дейін, бірінші қосымша жіп – 10-нан 19-ға дейін, ал екінші қосымша жіп –20-дан 2-ға дейін сандарды шығарсын (олардың жұмысы осындай).

Жіптер синхрондалмаған, жаһандық айнымалының көмегімен синхрондалған, бір үдеріс жіптерінің синхрондалу қасиеттірі - **Sleep, Join**и **lock** әдістерінің көмегімен синхрондалған бірнеші оқу бағдарламасын қарастырайық.

Синхрондалмаған жіптермен бағдарламада бірінші жіп өз сыныбының ішінде хабардар етілген, ал екінші жіп – статикалық әдіспен көрсетілген.

10.3.1 Жаһандық айнымалыны пайдалану

Flag жаһандық айнымалының көмегімен жіптердің синхрондалу мысалын қарастырайық. Барлық жіптер жұмысқа қосылады, бірақ жіптердің «денелері» жаһандық айнымалыдан жіп жұмысына рұқсат «тосатын» **goto** және **if** операторларымен блокталынады.

10.3.2 Sleep әдісін пайдалану

Ағымдағы жіпті миллисекундтардың берілген санына блоктайтын Thread.Sleep() әдісінің көмегімен жіптердің синхрондалу мысалын қарастырайық.

Thread.Sleep(0) әдісі – бөлінген уақыт квантынан бас тартады.

Thread.Sleep(100) әдісі –100 миллисекундқа жіптің орындалуын блоктайды –» 100 миллисекундқа «ұйықтау».

Жіп блоктаушы конструкцияларын пайдаланған соң блоктау аяқталғанша дейін уақыт кванттарының алуын тоқтататынын атап кету керек.

Мысалда бағдарламаның консолдық терезесіне әр шығудан кейін жіптердің блокталуы жүзеге асырылған.

10.3.3 Join әдісін пайдалану

Ағымдағы жіптің аяқталуына дейін шақырушы жіпті блоктайтын Join() әдісінің көмегімен жіптерді синхрондаудың мысалын қарастырамыз. Сонымен, басқа жіп аяқталғанша дейін Join() әдісімен блоктап тастауға болады.

Join() әдісн пайдаланудың үш нұсқасы бар:

Бірінші жағдайда бірінші жіп екінші және үшінші жіптердің орындалуын «блоктайды», олар блоктау аяқталған соң синхронды емес болып жұмыс істейді.

Екінші жағдайда тек бастапқы жіп «блокталынады», ал бірінші және екінші жіптер синхронды емес болып жұмыс істейді.

Үшінші жағдайда алғашқысында бірінші жіп екінші және бастапқы жіптердің орындалуын «блоктайды», бұдан соң екінші жіп басты жіпті «блоктайды».

Блоктаудың осы құралы жіптердің жұмысын ретке келтіруге мүмкіндік береді.

10.3.4 **lock** операторын пайдалану

Lock операторы **Mutex** және **Semaphore** синхрондау объектілерімен бірге арнайы блоктаушы конструкцияларға жатады**.**

Әртүрлі үдерістердің жіптерін блоктайтын **Mutex** және **Semaphore** синхрондау объектілеріне қарағанда, **lock** операторы тек бір үдерістің ішінде ғана жіптердің жұмысын блоктауға мүмкіндік береді. Сондықтанда **lock** операторының жұмысын осы дәрісте қарастырамыз.

Алдыңғы дәрісте біз жіптерге деректерді тапсыру үшін **lock** операторының жұмысын қарастырған болатынбыз. Осы мысалда y = a*x + b/x + c өрнегін есептеу үшін үш жіпті қосамыз. Бірінші жіп c мәнін алады және тиісінше екінші және үшінші жіптерде орындалатын a*xи b/x есептеулердің нәтижелерін тосады. x айнымалысы бағдарламада жаһандық айнымалы ретінде хабардар етіледі.

Оқу мақсаттарында бүкіл бағдарламаның жүзеге асырылуын 4 файл ретінде орындаймыз – бағдарлама файлы және әдістері үшін жіптер қосылатын болатын сыныптардың үш файлы. Бағдарламаның бірінші файлының бастапқы коды келесідей түрде болады:

Жобаға Klass_1 файлын құрастыру және қосу үшін Project режімінде AddClass пәрменін таңдап алу және пайда болған терезеде Klass_1 деген құрастырылатын файлдың атауын көрсету қажет.

Орта Klass_1.cs файлын құрастырады, оны жобаға қосады және файл бағдарламасының кодын редакциялау режіміне ауыстырады (10.1 суретін қараңыз).

Klass_1.cs файлында диалог режіміне **c** айнымалысының мәнін енгіземіз және жұмыс нәтижесін бүкіл өрнекті есептеу үшін пайдаланатын болатын жұмысқа жобаның екінші жібін қосамыз.

Кодтан бірінші жіптің жұмысын аяқтау үшін екінші жіп жұмысының (екінші сыныптың әдісімен жұмыс істейтін) нәтижесі қажетті екенін көреміз. Klass_2.cs файлын құрастырамыз және оны жобаға қосамыз.

Klass_2.cs файлында диалог режіміне b айнымалысының мәнін енгіземіз, b/x есептейміз және жұмысқа жобаның үшінші жібін қосамыз, оның жұмыс нәтижесін бүкіл өрнекті есептеу үшін пайдаланамыз.

Klass_2.cs файлы келесі бастапқы кодқа ие:

Klass_3.cs файлында диалог режіміне a айнымалысының мәнін енгіземіз, a * x есептейміз және жобаның үшінші жібінің жұмысын аяқтаймыз. Оның жұмысының аралық нәтижесін монитор экранына шығарамыз.

Бағдарлама жұмысының нәтижелері бойынша x айнымалысының мәні енгізілген соң бірінші жіп қосылатыны, бұдан соң екінші жіп және кейіннен үшінші жіп қосылатынын көреміз. Жіптер жұмысының аяқталуы кері ретпен жүргізіледі. Жіптер жұмысының синхрондалуы lock операторының көмегімен және алдыңғы жіптің жіптік функциясынан кезекті жіпті қосумен қол жеткізіледі.