Дәріс 7. Параллельді есептеу: блоктау және ашу

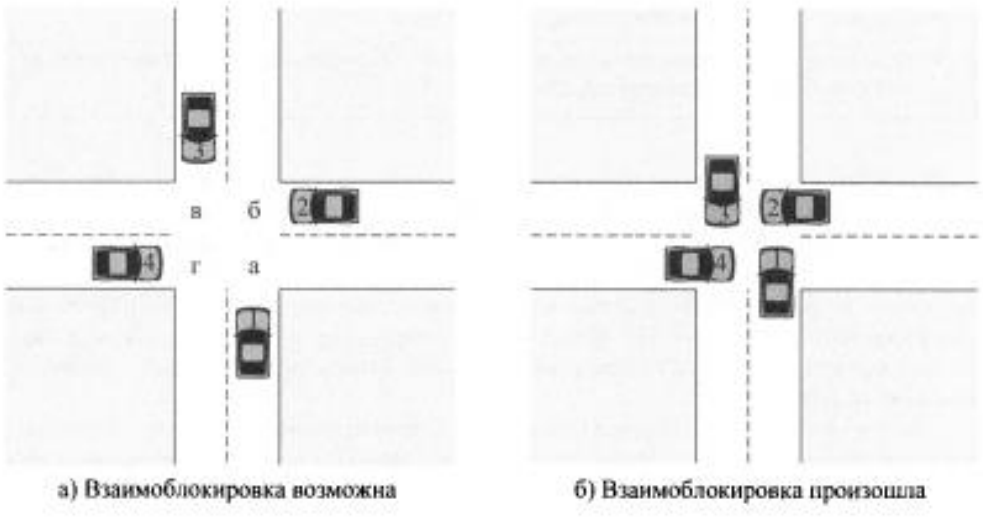
7.1. Өзара блоктау принциптері
7.2. Блоктаудың алдын алу
7.3. Блоктауды жою
7.4. Блоктауды анықтау
7.5. Философтардың тамақтану туралы есеп

7.1. Өзара блоктау принциптері

Өзара блоктау (deadlock) жүйелік ресурстар үшін күресте бәсекелесетін немесе бір-бірімен байланысатын көптеген процестерді тұрақты бұғаттау ретінде анықталуы мүмкін. Егер жиынтықтың әр процесі оқиғаның күтілуіне кедергі келтірсе (әдетте кейбір сұралған ресурстың босатылуы), онда жиынтықтың басқа бұғатталған процесі ғана туындауы мүмкін.

Ең жарқын мысал-көлік блоктауы. 7.1- суретте төрт автомобиль қиылысты бір уақытта кесіп өтуі керек жағдай көрсетілген. Қиылыстың төрт квадранты-бұл процестер қажет ететін ресурстар. Атап айтқанда, барлық төрт көліктің қиылысын сәтті кесіп өту үшін қажетті ресурстар келесідей.

- Солтүстікке қарай жүретін 1 автомобильге А және В квадраттары қажет.
- Батысқа қарай қозғалатын 2 автомобильге В және В квадранттары қажет.
- Оңтүстікке қарай қозғалатын 3 автомобильге В және Г квадранттары қажет.
- Шығысқа қарай қозғалатын 4 автомобильге G және А квадранттары қажет.



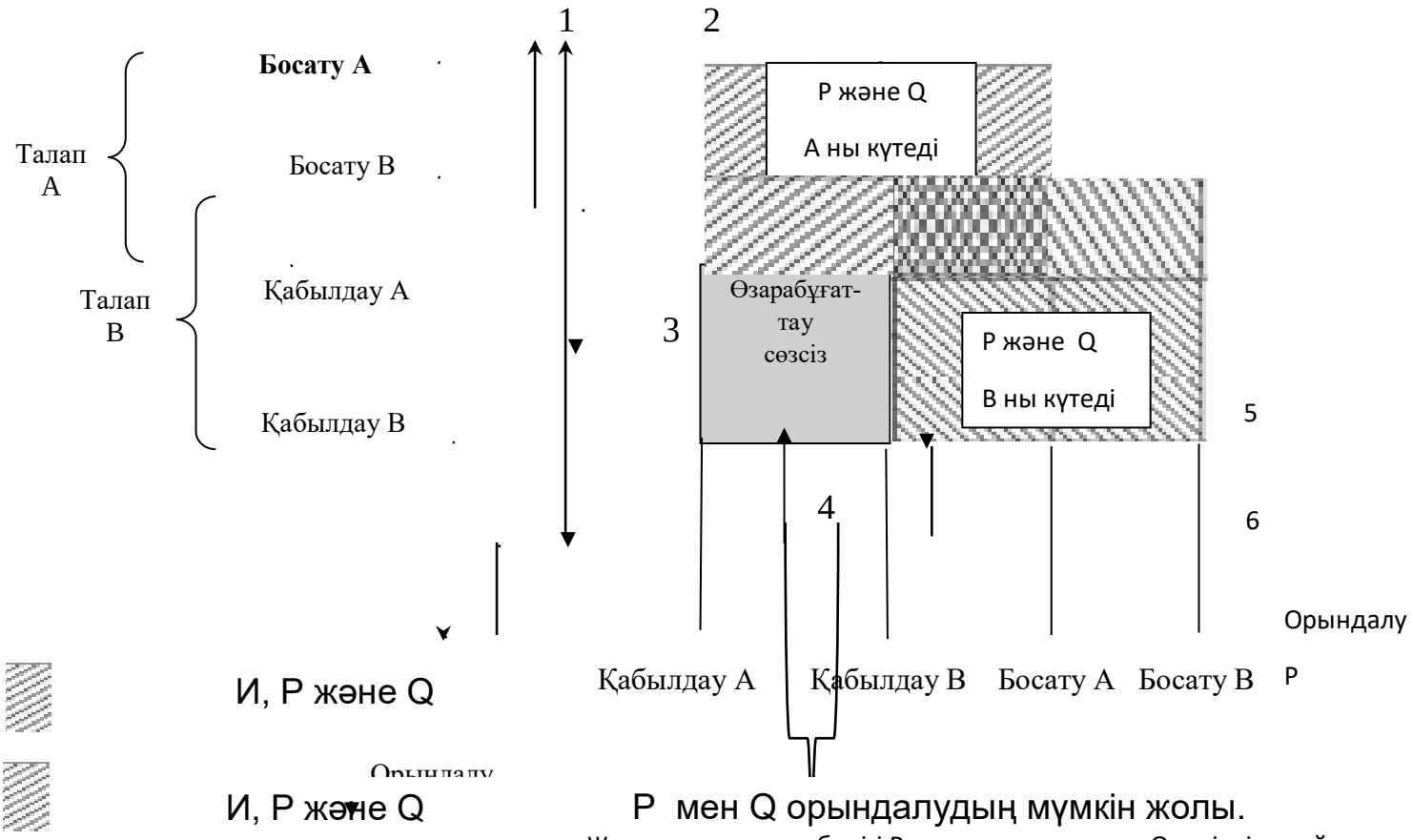
7.1-сурет. Бекіту мысалы

Қиылысты кесіп өту ережесі-автомобиль оң жақта жүргенге жол беруі керек. Егер қиылысты бір уақытта төрт автомобиль кесіп өтсе, олардың әрқайсысы Ережеге сәйкес қиылысқа кіруден бас тартса, құлыптау пайда болады. Бұл барлық төрт автомобиль қиылысқа кірген жағдайда да пайда болады, өйткені әр автомобиль бір ресурсты (бір квадрантты) алады және басқа көлік қиылысты кесіп өту үшін қажет келесі квадрантты босатуды күтіп тұрақта қалады.

Енді процестер мен компьютерлік ресурстардың қатысуымен блоктау суретін қарастырыңыз. 7.2 – суретте екі A және B ресурстары үшін күресте бәсекелесетін екі процестің бірлескен орындалу диаграммасы (бірлескен прогрессиялық диаграмма) көрсетілген. P және Q процестері жалпы келесі көрініске ие:

Процесс P	Процесс Q
•••	•••
Алу A	Алу B
•••	•••
Алу B	Алу A
•••	•••
Босату A	Босату B
•••	•••
Босату B	Босату A
•••	•••

7.2-суретте X осі P процесінің орындалуын, ал y осі Q процесінің орындалуын білдіреді, сондықтан екі процестің бірлескен орындалуы шығу тегінен солтүстік-шығыс бағытта жүреді.



Р мен Q орындалудың мүмкін жолы.

Жолдың көлденең бөлігі Р орындалуына және Q күтілуіне сәйкес келеді.

Жолдың тік бөлігі Q орындалуына және Р күтілуіне сәйкес келеді.

7.2-сурет. Тұйықталу мысалы

7.2-суретте процестерді орындаудың алты түрлі жолы көрсетілген.

1. Q - В ресурсын, содан кейін А ресурсын алады, содан кейін В және А ресурстарын шығарады. Р процесі орындалуды жалғастыра отырып, ол екі ресурсты да ала алады.

2. Q - В ресурсын, содан кейін А ресурсын алады. Р процесі А ресурсын сұраған кезде басталады және блоктайды. Q - В және А ресурстарын шығарады. Р процесі жалғасып жатқанда, ол екі ресурсты да ала алады.

3. Q - В ресурсын алады, содан кейін Р - А ресурсын алады. Тұйықталу сөзсіз, себебі Q - А ресурсы сұралған кезде блоктайды, ал В ресурсы сұралғанда Р процесі блоктайды.

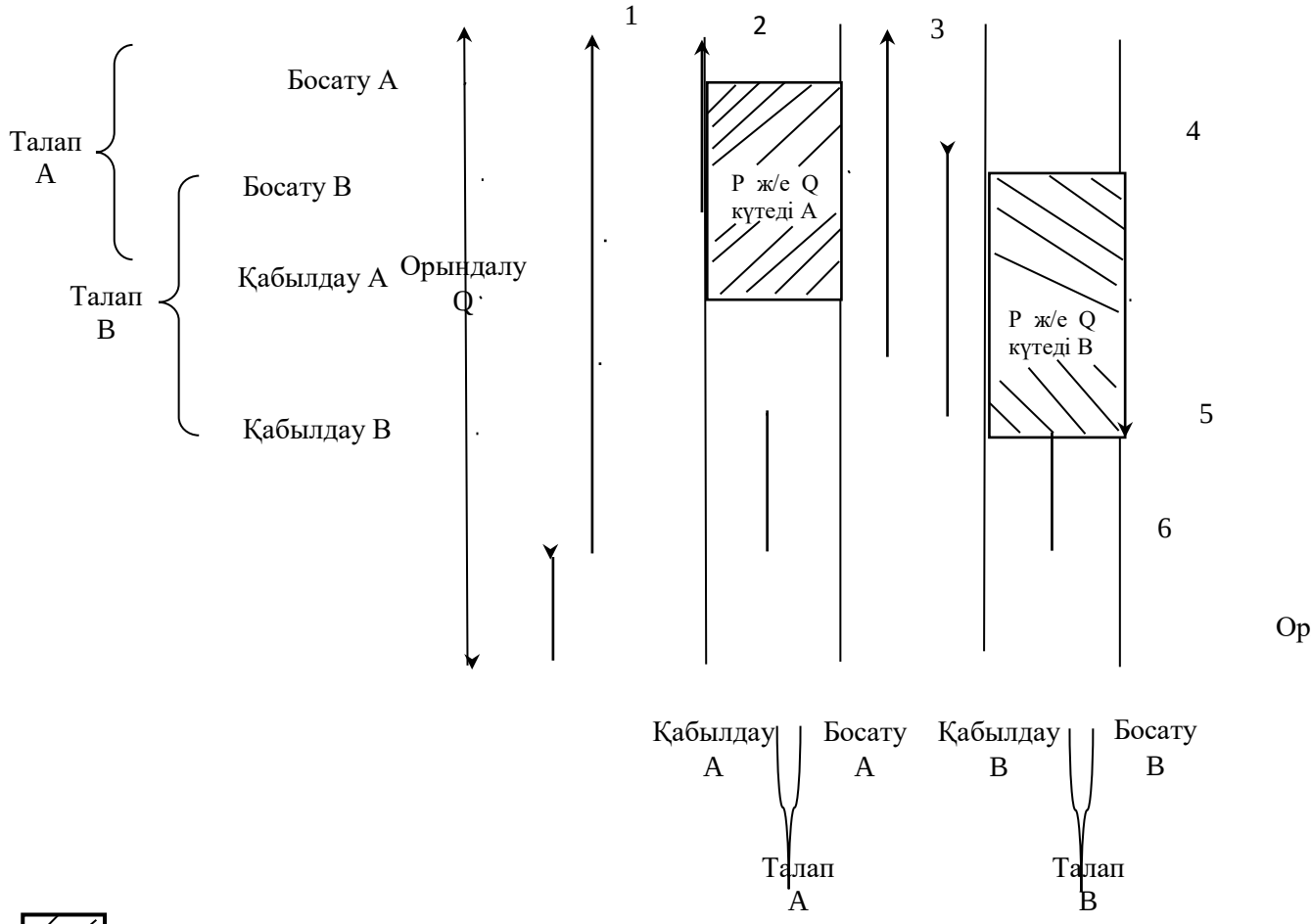
4. Р - А ресурсын алады, содан кейін Q - В ресурсын алады. Тұйыққа тірелуі сөзсіз, себебі Q - А ресурсы сұралған кезде блоктайды, ал В ресурсы сұралғанда Р процесі блоктайды.

5. P - A ресурсын, содан кейін B ресурсын алады. Q процесі B ресурсын сұраған кезде басталады және блоктайды. P - A және B ресурстарын шығарады. Q процесі жалғасқанда, ол екі ресурсты да ала алады.

6. P - A ресурсын, содан кейін B ресурсын алады, содан кейін A және B ресурстарын шығарады. Q процесі іске қосылуын жалғастырғанда, ол екі ресурсты да ала алады.

7.2-суретте фатальды аймақ бар - егер орындау жолы осы аймаққа енсе, тығырықтан шығу сөзсіз. Тұйықтың орын алуы немесе болмауы процестердің орындалу динамикасына және қолданбаның қалай құрастырылғанына байланысты. Дегенмен, екі процесс өлімге әкелетін аймаққа кіретін жолды жасау үшін бірге жұмыс істегенде ғана тығырықтан шығу сөзсіз.

Мысалы, P процесі екі ресурстарды бір уақытта алуды қажет етпейді делік. Бұл жағдай 7.3 - суретте көрсетілгендей: екі процесс бір-біріне қатысты қалай орындалғанына қарамастан, блоктау мүмкін емес.



P ж/е Q орындалуының мүмкін жолы.

Жолдың көлденең бөлігі P орындалуына ж/е Q күтуіне сәйкес.

Жолдың тік бөлігі Q

7.3-сурет Бекітудің болмауының мысалы [15]

Қайта пайдаланылатын ресурстар

Ресурстарды екі негізгі санатқа бөлуге болады: қайта пайдаланылатын (reusable) және тұтынылатын (consumable). Қайта пайдаланылатын ресурстарды бір уақытта тек бір процесте қауіпсіз пайдалануға болады және таусылмайды. Қайта пайдаланылатын ресурстардың мысалдары-процессор, енгізу-шығару арналары, негізгі және екінші жад, перифериялық құрылғылар, сонымен қатар файлдар, мәліметтер базасы және семафорлар сияқты мәліметтер құрылымы.

Қайта пайдаланылатын ресурсты блоктаудың мысалы ретінде біз D диск файлына және T ағынына ерекше қол жеткізу үшін бәсекелесетін екі процесті қарастырамыз, бағдарлама 7.4-суретте көрсетілген операцияларды орындайды.

P процессі		Q процессі	
Қадам	Әрекет	Қадам	Әрекет
P0	Сұрау(D)		
P1	Құлыптау(D)		
P2	Сұрау(T)		
P3	Құлыптау(T)		
P4	Функцияны орындау		
P5	Бұғаттан шығару(D)		
P6	Бұғаттан шығару(T)		
		q0	Сұрау(T)
		q1	Құлыптау(T)
		q2	Сұрау(D)
		q3	Құлыптау(D)
		q4	Функцияны орындау
		q5	Бұғаттан шығару(T)
		q6	Бұғаттан шығару(D)

7.4-сурет Қайта пайдаланылатын ресурс үшін күрестегі екі процестің бәсекелестік мысалы

Блоктау әр процесс бір ресурсты ұстап, екіншісін сұраған кезде жүзеге асырылады. Мысалы, блоктау екі процестің келесі ауысуында болады: p0, p1, q0, q1, p2, q2. Осындай блоктармен жұмыс істеу кезіндегі стратегиялардың бірі ресурстарды сұрату тәртібіне жүйелік шектеулер қою болып табылады.

Тұтынылатын ресурстар

Шығын материалдары – жасалуы (өндірілуі) және жойылуы (тұтынылуы) мүмкін ресурстар. Әдетте тұтынылатын белгілі бір түрдегі ресурстардың көлеміне шектеу қойылмайды. Тұтынылатын ресурстардың мысалдары үзілістер, сигналдар, хабарламалар және енгізу/шығару буферлеріндегі ақпарат болып табылады. Шығарылатын ресурстары бар тығырықтан шығудың мысалы ретінде әрқайсысы басқа процесстен хабар алуға тырысатын, содан кейін оған хабарлама жіберетін келесі процестер жұбын қарастырыңыз.

Процесс P1	Процесс P2
•••	•••
Receive(P2);	Receive(Pl);
•••	•••
Send(P2, Ml) ;	Send(P_l, M2) ;

Егер Receive операциясы бұғаттағыш болса (яғни, қабылдау процесі хабарлама алынғанға дейін бұғатталады), блоктау жүзеге асырылады.

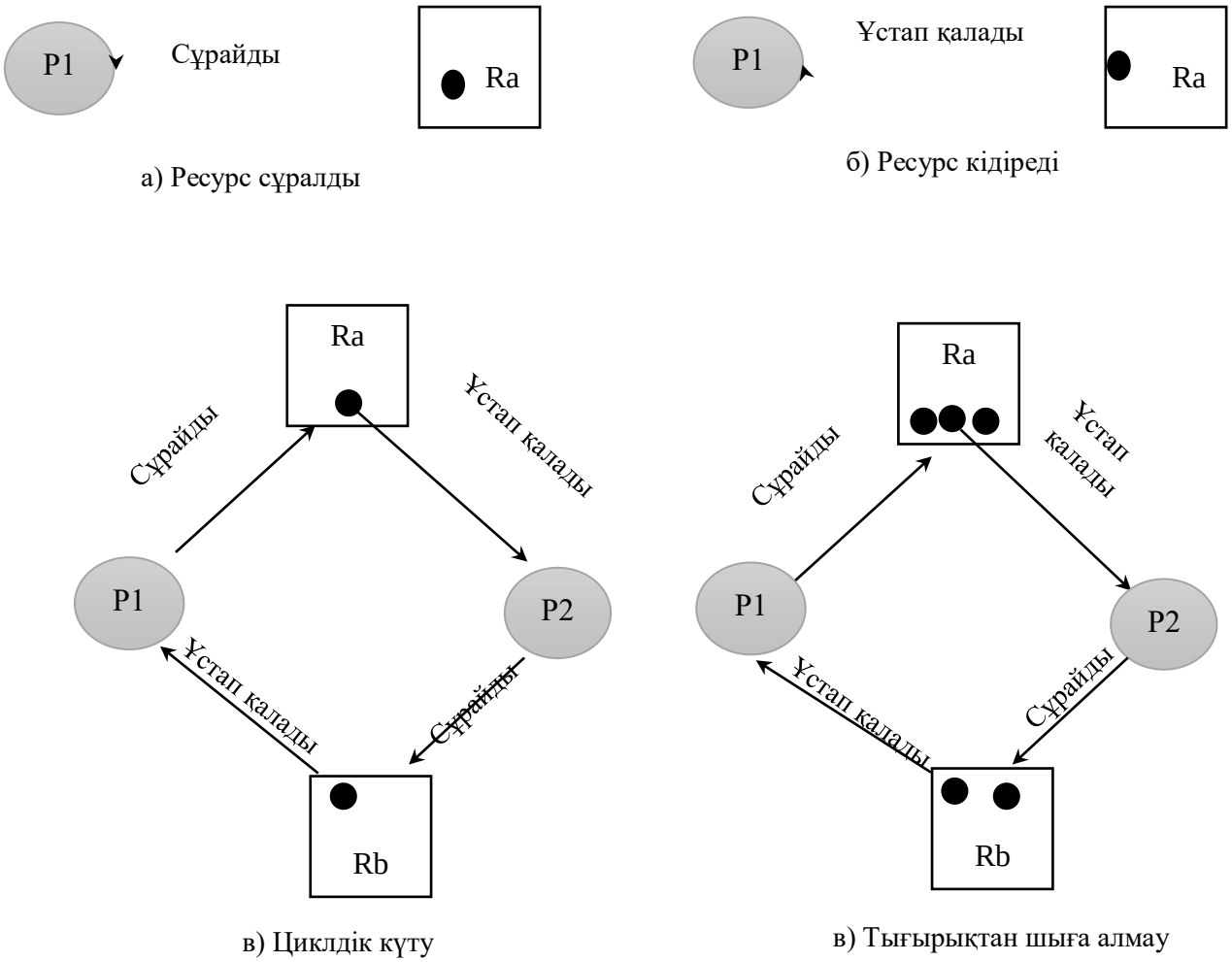
Бекітудің барлық түрлерімен жұмыс істеудің бірыңғай тиімді стратегиясы жоқ. Бұл мәселені шешудің негізгі тәсілдері келесідей.

- Бұғаттаудың алдын алу. Блоктаудың пайда болуының қажетті үш шартының бірін орындауға немесе циклдік күту жағдайының пайда болуына жол бермеңіз.
- Блоктарды жою. Ресурс сұрауын орындамаңыз, егер оны таңдау бұғаттауға әкелуі мүмкін.
- Блоктауды анықтау. Мүмкіндігінше ресурстар сұрауларын орындаңыз, бірақ мезгіл-мезгіл құлыптаудың болуын тексеріп, қалпына келтіру шараларын қолданыңыз.

Ресурстарды бөлу бағандары

Процестер арасында ресурстарды бөлуді сипаттау үшін пайдалы құрал Холт (Holt) ұсынған ресурстарды бөлу графигі болып табылады [109]. Ресурстарды бөлу графигі-бұл әр процесс және әр ресурс түйіндермен ұсынылатын ресурстар мен процестер жүйесінің күйін бейнелейтін бағытталған график. Процестен ресурсқа бағытталған графикалық доғалар процесс сұраған, бірақ оған әлі берілмеген ресурсты көрсетеді (сурет. 7.5, а). Ресурс түйінінде ресурстың әр данасы нүктемен көрсетілген. Бірнеше даналары болуы мүмкін ресурс түрлерінің мысалдары операциялық жүйедегі

ресурстарды басқару модулі арқылы бөлінген енгізу/шығару құрылғылары болып табылады. Қайта пайдалануға болатын ресурс түйінінен процеске бағытталған график доғасы орындалған сұранысты көрсетеді (7.5-сурет, б), яғни, бұл процеске бір ресурс бірлігі тағайындалды. Тұтынылатын ресурс түйінінен процеске бағытталған график доғасы процестің сол ресурс өндірушісі екенін көрсетеді. Суретте. 7.5c тығырықтан шығудың мысалын көрсетеді. Суретте. 7.5d Тұйықталу жоқ, себебі әрбір ресурстың бірнеше бірлігі қолжетімді.



7.5-сурет. Ресурстарды бөлу графиктерінің мысалдары

Тұйық жағдайлар

Тығырықтан шығу үшін үш шарт қажет.

1. Бір-бірін жоққа шығару. Бір уақытта тек бір процесс ресурсты пайдалана алады.

2. Ұстаңыз және күтіңіз. Басқа ресурстарды күту кезінде процесс бөлінген ресурстарды ұстай алады.

3. Қайта бөлу жоқ. Ресурсты ұстап тұрған процестен мәжбүрлеп алу мүмкін емес.

Бұл шарттар жиі орындалады. Мысалы, нәтижелер мен мәліметтер базасының тұтастығына кепілдік беру үшін өзара анықтама қажет. Сол сияқты, қайта бөлуді ерікті түрде қолдану мүмкін емес, әсіресе кері қайтару механизмін қамтамасыз ету қажет болған кезде мәліметтермен жұмыс жасау кезінде.

Жоғарыда аталған үш шарттан басқа, нақты бекіту үшін қажетті, бірақ жеткіліксіз, төртінші шарт қажет.

4. Циклдік күту. Жабық процестер тізбегі бар, олардың әрқайсысы берілгеннен кейін тізбекте келесі процесске қажет кем дегенде бір ресурсты ұстайды (7.5, в -суретті қараңыз). Жоғарыда аталған төрт Шарттың жиынтығы-бұл бекітудің қажетті және жеткілікті шарты.

Блоктау мәселелерін шешу үшін үш негізгі тәсіл бар. Біріншіден, шарттардың бірін жоятын стратегияны қабылдау арқылы бұғаттаудың алдын алуға болады (1-4 шарттар). Екіншіден, ресурстарды бөлудің қазіргі жағдайына негізделген тиісті динамикалық таңдау жасау арқылы бұғаттауды жоюға болады. Үшіншіден, сіз бұғаттауды анықтауға тырысуға болады (1-4 шарттар) және жұмыс қабілеттілігін қалпына келтіру үшін шаралар қолданыңыз.

7.2. Блоктаудың алдын алу

Стратегия болдырмау, шын мәнінде, білдіреді, осындай жүйесін әзірлеуге мүмкіндік береді деген сөз алып тасталсын өзін мүмкіндігі взаимоблокировок. Бекітудің алдын алу әдістерін екі сыныпқа бөлуге болады. Жанама әдіс-бұғаттаудың пайда болуының алғашқы үш шартының бірін болдырмау; Тікелей әдіс циклдік күтуге жол бермейді (4 шарт). Әр жағдайға байланысты әдістерді бөлек қарастырыңыз.

Өзара тұжырымдар

Жалпы жағдайда өзара тұжырымдарды пайдаланудан аулақ болу мүмкін емес. Егер ресурсқа қол жеткізу ерекше болуы керек болса, онда операциялық жүйе өзара алып тастауды қолдауы керек. Файлдар сияқты кейбір ресурстар оқуға бірнеше рет қол жеткізуге және жазуға ерекше қол жеткізуге мүмкіндік береді. Бірақ бұл жағдайда да, егер Файлға жазу құқығы бір уақытта бірнеше процесті қажет етсе, блоктау пайда болуы мүмкін.

Ұстау және күту

Бұл шартты процестің барлық қажетті ресурстарды бір уақытта сұрауын талап ету арқылы болдырмауға болады және мұндай сұрау бір уақытта толық орындалмайынша процесті бұғаттауға болады. Бұл тәсіл екі себепке байланысты тиімсіз. Біріншіден, процесс барлық талап етілген ресурстардың бір уақытта қол жетімділігін ұзақ уақыт күте алады, ал іс жүзінде ол олардың бір бөлігімен ғана жұмыс істей алады. Екіншіден, процесс талап ететін ресурстар айтарлықтай уақыт бойы пайдаланылмай қалуы мүмкін, оның

барысында олар басқа процестерге қол жетімді емес. Тағы бір мәселе-бұл процесс оған қандай ресурстарды қажет ететінін алдын-ала білмеуі мүмкін.

Қайта бөлудің болмауы

Бұл жағдайды бірнеше жолмен болдырмауға болады. Мысалы, сіз мұны жасай аласыз: егер процесс кейбір ресурстарды ұстап қалса және оған келесі сұраудан бас тартса, онда ол басып алынған ресурстарды босатып, қажет болған жағдайда оларға қол жеткізуден бас тартылған ресурстармен бірге қайтадан сұрауы керек. Екінші жағынан, егер процесс белгілі бір ресурстарды қажет етсе, онда басқа процесс басып алған болса, онда операциялық жүйе бұл процесті қысып, одан алынған ресурстарды босатуды талап етуі мүмкін. Бұл әдіс барлық процестер әртүрлі басымдықтарға ие болған жағдайда ғана бұғаттаудың алдын алады.

Циклдік күту

Циклдік күту жағдайларын ресурстардың түрлерін ретке келтіру арқылы болдырмауға болады. Сонымен қатар, егер процесс R типті ресурсты сұраса, онда ол тек r-ге сәйкес көрсетілген тәртіпке сәйкес ресурстарды сұрай алады, ұстап қалу мен күтудің алдын алу жағдайында циклдік күтудің алдын алу технологиясы тиімсіз болуы мүмкін, бұл процестің жылдамдығын төмендетеді және ресурстарға қол жетімділікті қажет етпейді.

7.3. Блоктауды жою

Блоктау мәселесін шешудің тағы бір тәсілі-блоктауды жою. Бекітулердің алдын алу жағдайында біз ресурстардың сұраныстарына белгілі бір шектеулер қойдық, осылайша бекітулердің болуы үшін қажетті жағдайлардың кем дегенде біреуін жүзеге асыруды мүмкін Етпейміз және сол арқылы олардың пайда болу мүмкіндігін болдырмаймыз. Өкінішке орай, бұл әдіс ресурстарды тиімсіз пайдалануға және процестің жылдамдығын төмендетуге әкеледі. Құлыптарды жою құлыптардың пайда болуына үш қажетті жағдайдың болуына мүмкіндік береді, бірақ біз процестерді өзара оқшаулау жағдайына қол жеткізе алмайтындай шаралар қабылдаймыз. Тиісінше, блоктауды жою алдын-алуға қарағанда есептеудің параллелизмін қамтамасыз етеді. Ресурстың ағымдағы сұранысы қанағаттандырылған жағдайда өзара бұғаттауға әкелуі мүмкін бе деген шешім бұл жағдайда динамикалық түрде қабылданады. Бұл бөлімде біз құлыптарды жоюдың екі тәсілімен танысамыз.

1. Егер оның сұраулары бұғаттауға әкелуі мүмкін болса, процесті бастамаңыз.
2. Егер оларды орындау бұғаттауға әкелуі мүмкін болса, процестің қажеттіліктерін қанағаттандырмаңыз.

Процесті бастауға тыйым салу

Ресурстардың әр түрлі түрлерінің N процестері мен m жүйесін қарастырыңыз. Біз келесі векторлар мен матрицаларды анықтаймыз.

Ресурс: $\mathbf{R} = (R_1, R_2, \ldots, R_m)$	Общее количество каждого ресурса в системе
Доступность: $\mathbf{V} = (V_1, V_2, \ldots, V_m)$	Общее количество каждого ресурса, не выделенного процессам
Требования: $\mathbf{C} = \begin{pmatrix} C_{11} & C_{12} & \cdots & C_{1m} \\ C_{21} & C_{22} & \cdots & C_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ C_{n1} & C_{n2} & \cdots & C_{nm} \end{pmatrix}$	C_{ij} — запрос процессом i ресурса j
Распределение: $\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix}$	A_{ij} — текущее распределение процессу i ресурса j

Ресурс:R=(R_1, R_2,....., R_m)	Жүйедегі әр ресурстың саны
Қолжетімділік: V=(V_1,V_2,….,V_m)	Процессте бөлінбеген, ресурстардың жалпы саны
Талаптар:C=	C_{ij}- і ресурс процесі арқылы сұрау j
Тарату:A=	A_{ij}- і ресурс процесіне ағымдағы бөлу j

Әр жол процестердің біреуін сипаттайтын талап матрицасы әр процестің әр түрлі ресурстарға қойылатын максималды талаптарын көрсетеді, яғни C_{ij}- бұл j ресурсының і процесінің талаптары. Сол сияқты, A_{ij-i} процесіне бөлінген j ресурсының ағымдағы мөлшері.

Келесі қатынастар орындалуы керек.

1. $R_j = V_j + \sum_{i=1}^{n} \quad A_{ij}$ барлық j үшін: барлық ресурстар бос немесе бөлінген.
2. $C_{ij} \leq R_j$ барлық і және j үшін: ешқандай процесс жүйеде оның жалпы санынан асатын ресурсты қажет ете алмайды.
3. $A_{ij} \leq C_{ij}$ барлық і және j үшін: ешқандай процесс талап етілгеннен көп ресурстар ала алмайды.

Барлық көрсетілген мәндер анықталған кезде, егер оның ресурстық талаптары бұғаттауға әкелуі мүмкін болса, Жаңа процесті бастауға тыйым салатын блоктарды жою стратегиясын жасай аламыз. Жаңа $P_{(n+1)}$ процесі тек іске қосылады.

$R_j \geq C_{(n+1)j} + \sum_{i=1}^{n} \quad C_{ij}$ барлық j үшін.

Бұл дегеніміз, жаңа процесті бастау барлық ағымдағы процестердің максималды талаптары мен іске қосылған процестің талаптары қанағаттандырылған жағдайда ғана болады. Бұл стратегия ешқандай жағдайда

оңтайлы емес, өйткені ол ең нашар нәрсені болжайды: барлық процестер бір уақытта максималды талаптарды қояды.

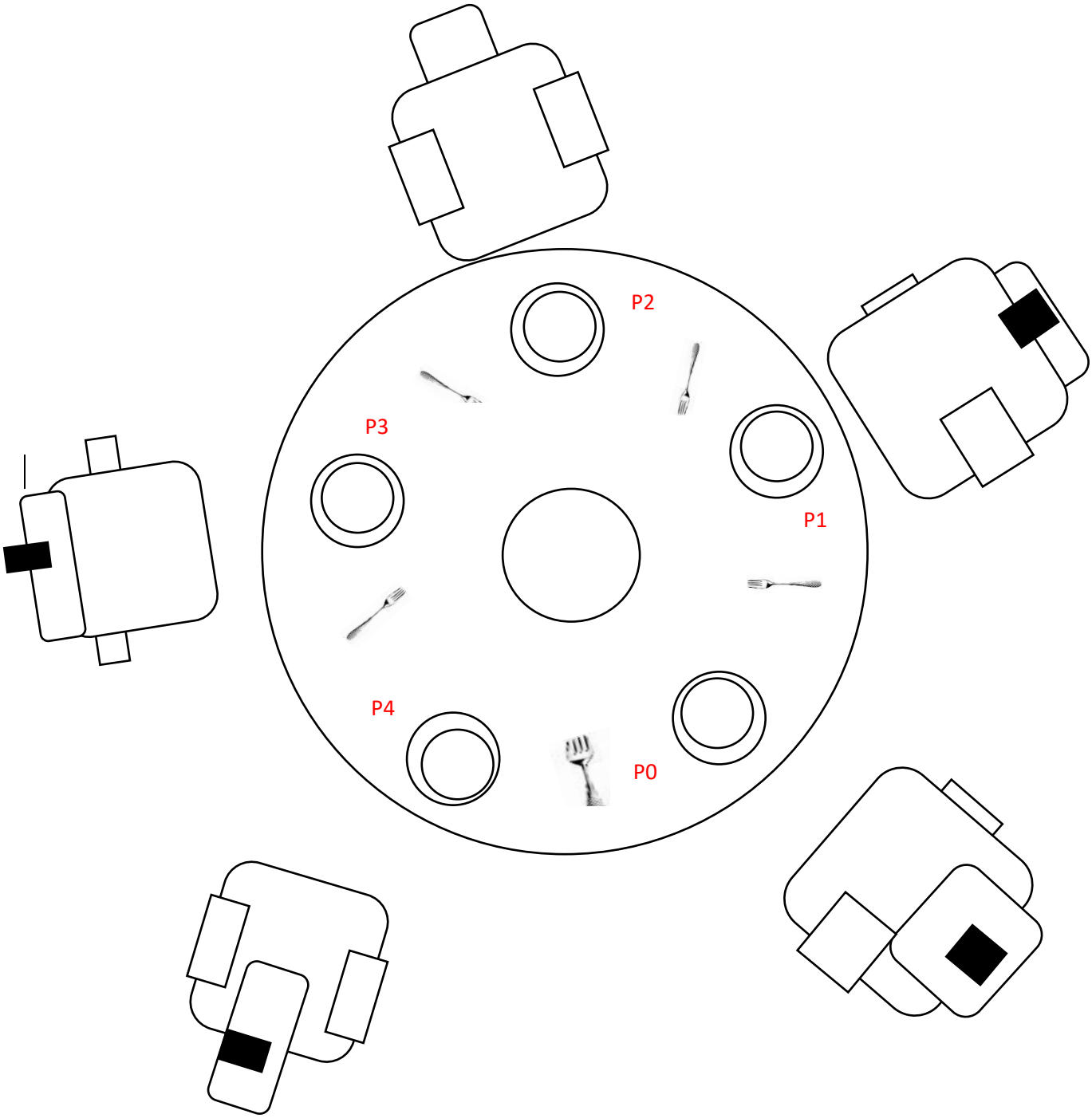
Ресурсты бөлуге тыйым салу

Банкир алгоритмі деп аталатын ресурстарды бөлуге тыйым салу стратегиясы алғаш рет жұмыста ұсынылды [66]. Біз қарауды күй (*state*) және қауіпсіз күй (safe state) тұжырымдамаларынан бастаймыз. Процестер мен ресурстардың белгіленген саны бар жүйеге тоқталайық. Уақыттың әр сәтінде процесс оған бөлінген бірнеше ресурстарға ие болуы мүмкін (немесе жоқ). Жүйенің күйі-бұл ресурстарды процестер бойынша ағымдағы бөлу. Сондықтан күйді бұрын анықталған екі вектор (ресурстар мен қол жетімділік) және екі матрица (талаптар мен бөлу) ретінде ұсынуға болады.

Қауіпсіз күй - бұл кем дегенде бір тізбегі бар, ол бұғаттауға әкелмейді (яғни, барлық процестер аяқталғанға дейін орындалуы мүмкін). Қауіпсіз емес жағдай сәйкесінше қауіпті жағдай деп аталады.

7.5. Философтардың тамақтану туралы есеп

Енді [65] Дейкстра ұсынған «Дин-философтар мәселесін» қарастырайық. Бес философ бірге өмір сүрген. Олар бір дөңгелек үстелде тамақтанды (7.10-сурет), оның үстіне спагеттиден жасалған үлкен ыдыс, бес тәрелке, әр философқа бір-бірден және бес шанышқы қойылған. Аш философ үстел басына отырады да, екі шанышқыны пайдаланып, тамақ іше бастайды. Мәселе бір-бірін жоққа шығаруды (екі философ бір шанышқыны бір уақытта пайдалана алмайды) қамтамасыз ететін және тығырықтан шығуға және аштыққа жол бермейтін кешкі ас рәсімін (алгоритмін) жасау болып табылады. Динингг философтарының мәселесі ортақ ресурстармен жұмыс істеу кезінде көп ағынды қосымшаларда пайда болатын және сәйкесінше синхрондау мәселесіне жаңа тәсілдерді әзірлеу кезінде сынақ ретінде әрекет ете алатын типтік мәселе ретінде қарастырылуы мүмкін.



7.10-сурет. Философтар асханасы

Семафорларды қолдану арқылы шешу

7.11-суретте семафорлардың көмегімен бұл мәселені шешу ұсынылады. Үстел басына отырған әрбір философ алдымен сол шанышқыны, содан кейін оң жақ шанышқыны алады. Философ түскі асты ішкеннен кейін, ол қолданған шанышқылар ауыстырылады. Бірақ мұндай шешім тығырыққа апаруы мүмкін, егер философтар бір мезгілде аш болса, барлығы үстелге отырып, сол жақта жатқан шанышқыларды бір уақытта алса. Бұл жағымсыз жағдайда олар аштыққа ұшырайды.

ҚОЛДАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1. Garg, R.; Verma, G.Operating Systems [OP]: An Introduction - Softcover Publisher: Mercury Learning & Information, 2017. 290 p.
2. https://gifer.com/ru/7h0m
3. https://3dnews.ru/1034959
4. Darrell Hajek, Cesar Herrera, Flor Narciso Principles of Operating Systems. Independently Published (24 April 2020) 176 pages.
5. Andrew S. Tanenbaum and Herbert Bos. Modern Operating Systems. 4/E.1136 pages, Pearson India, 2016.
6. Silberschatz Abraham, Galvin Peter Baer and Gadne Greg. Operating system concepts.
7. Amdahl GM (1967) Validity of the single-processor approach to achieve large scale computing capabilities. AFIPS Joint Spring Conference Proceedings 30 (Atlantic City, NJ, Apr. 18–20), AFIPS Press, Reston VA, pp 483–485.
8. https://studfile.net/.
9. https://habr.com/ru/post/40227/.
10. wikimedia.org
11. wordpress.com
12. blackandwhitecomputer.blog
13. http://www-inst.eecs.berkeley.edu/~n252/paper/Amdahl.pdf.
14. encyclopedia2.thefreedictionary.com
15. linustechtips.com
16. youtube.com/watch?v=w3K1JklY6D4