ЛЕКЦИЯ № 14. Операциялық жүйелерді әзірлеу

Операциялық жүйенің жобасы сәтті болуы үшін әзірлеушілер не қалайтыны туралы нақты түсінікке ие болуы керек. Мақсат болмаған жағдайда кейінгі шешімдер қабылдау өте қиын. Бұл сұрақты түсінікті ету үшін екі бағдарламалау тіліне, PL/1 және C. PL/1 тілін IBM корпорациясы 1960 жылдары жасаған, өйткені Fortran мен COBOL-одновременно бір уақытта ұстап тұру және Algol-дің Fortran мен COBOL-дан гөрі жақсы екендігі туралы ғалымдардың күңкілдерін есту, төзгісіз болды. Сондықтан барлық бағдарламашылардың сұраныстарын қанағаттандыратын жаңа тіл құру үшін комитет құрылды: PL / 1. Бұл тілде FORTRAN тілінің кейбір ерекшеліктері, COBOL тілінің кейбір ерекшеліктері және Algol тілінің кейбір қасиеттері болды. Жоба сәтсіздікке ұшырады, өйткені оған бір тұжырымдама жетіспеді. Жоба бір-біріне қайшы келетін қасиеттер жиынтығы болды, сонымен қатар PL/1 тілі бағдарламаларды тиімді құрастыру үшін тым көлемді және ыңғайсыз болды.

Енді C тіліне назар аударыңыз, оны тек бір адам, Деннисом Ритчи, жүйелік бағдарламалаудың жалғыз мақсаты үшін жасаған. Оның жетістігі орасан зор болды және бұл Ричидің не қалайтынын, не қаламайтынын білетіндігімен түсіндірілмеді. Нәтижесінде, пайда болғаннан кейін ондаған жылдар өткен соң, бұл тіл әлі де кең таралған. Өз мақсаттарыңыз туралы нақты идеяның болуы шешуші болып табылады.

Операциялық жүйені жасаушылар не қалайды? Жауап әр жүйеде әр түрлі болатыны анық және ендірілген және серверлік жүйелер үшін әр түрлі болады. Әмбебап операциялық жүйелер үшін келесі төрт тармақ негізгі болып табылады:

1. Абстракциялардың анықтамасы.
2. Қарабайыр операцияларды қамтамасыз ету.
3. Оқшаулауды қамтамасыз ету.
4. Аппаратураны басқару.

Операциялық жүйенің ең маңызды, бірақ ең қиын міндеті-дұрыс абстракцияларды анықтау. Олардың кейбіреулері, мысалы, процестер мен файлдар ұзақ уақыт бойы қолданылып келеді, олар айқын көрінуі мүмкін. Басқалары, мысалы, орындау ағындары, жаңа, сондықтан онша қалыптаспаған ұғымдар. Мысалы, егер бірнеше ағыннан тұратын процесс болса, оның біреуі пернетақта кірісімен бұғатталған болса, онда жаңа процестегі ағын пернетақта кірісін де күтуі керек пе? Басқа абстракциялар синхрондауға, сигналға, жад моделіне, енгізу-шығару модельдеуіне және басқа салаларға қатысты.

Әрбір абстракция нақты деректер құрылымы ретінде жүзеге асырылуы мүмкін. Пайдаланушылар процестерді, файлдарды, арналарды және т.б. жасай алады. Мысалы, пайдаланушылар файлдарды оқып, жаза алады. Қарабайыр операциялар жүйелік қоңыраулар түрінде жүзеге асырылады. Пайдаланушының көзқарасы бойынша операциялық жүйенің жүрегі абстракциялармен қалыптасады және оларға операциялар жүйелік қоңыраулар арқылы мүмкін болады.

Бір компьютерде бір уақытта бірнеше пайдаланушы тіркеле алатындықтан, Операциялық жүйе оларды бір-бірінен ажырату механизмдерін қамтамасыз етуі керек. Бір қолданушы екіншісінің жұмысына араласпауы керек. Процестер тұжырымдамасы ресурстарды қорғау мақсатында топтастыру үшін кеңінен қолданылады.

Әдетте, файлдар мен басқа деректер құрылымдары қорғалады. Бөлу өте маңызды рөл атқаратын тағы бір орын-виртуализация: гипервизор виртуалды машиналардың басқа виртуалды машиналардың жұмысындағы барлық қиындықтардан оқшауланuын қамтамасыз етуі керек. Операциялық жүйені жобалаудың негізгі мақсаты-әрбір пайдаланушының қол жеткізу құқығы бар деректермен тек рұқсат етілген әрекеттерді орындай алуын қамтамасыз ету. Дегенмен, пайдаланушыларға деректер мен ресурстарды ортақ пайдалану қажет болуы мүмкін, сондықтан оқшаулау таңдамалы және пайдаланушылардың бақылауында болуы керек. Мұның бәрі операциялық жүйенің құрылғысын едәуір қиындатады. Электрондық пошта бағдарламалары веб-шолғыштарға зиян тигізбеуі керек. Бір ғана пайдаланушы болса да, әртүрлі процестерді бөлу керек. Процестерді бір-бірінен қорғау үшін Android сияқты кейбір жүйелерде бір пайдаланушыға тиесілі әрбір процесс басқа пайдаланушы идентификаторымен іске қосылады.

Интерфейсті дамыту

1-қағида: қарапайымдылық

Қарапайым интерфейсті түсіну және қатесіз жүзеге асыру оңайырақ. Барлық жүйені жасаушылар Француз ұшқышы және жазушысы Антуан де сент-Экзюперидің әйгілі дәйексөзін есте ұстауы керек: "кемелдікке енді қосатын ештеңе болмаған кезде емес, азайтатын ештеңе болмаған кезде қол жеткізіледі".

Бұл принцип кем дегенде операциялық жүйелерге қатысты азырақ жақсырақ екенін айтады. Басқаша айтқанда, бұл принципті Kiss (Keep it Simple, Stupid) аббревиатурасымен білдіруге болады, ол ойлау қабілетіне күмән келтіретін бағдарламашыға жүйені қиындатпауды ұсынады.

2-қағида: толықтығы

Әрине, интерфейс пайдаланушыларға қажет нәрсенің бәрін орындауға мүмкіндік беруі керек, яғни интерфейс толықтығына ие болуы керек. Осыған байланысты тағы бір дәйексөз ойға оралады, бұл жолы Альберт Эйнштейн айтқан сөз тіркесі: "бәрі мүмкіндігінше қарапайым болуы керек, бірақ оңай емес".

Басқаша айтқанда, операциялық жүйе одан талап етілетін нәрсені орындауы керек, бірақ одан артық емес. Егер пайдаланушыға деректерді сақтау қажет болса, Операциялық жүйе бұл үшін белгілі бір механизмді қамтамасыз етуі керек. Егер пайдаланушылар бір-бірімен байланысуы керек болса, Операциялық жүйе байланыс механизмін және т. б. қамтамасыз етуі керек. Ctss және MULTICS жүйелерін жасаушылардың бірі Фернандо Корбато turingaward сыйлығын алу туралы сөйлеген сөзінде қарапайымдылық пен толықтық ұғымдарын біріктіріп: "біріншіден, қарапайымдылық пен талғампаздықтың маңыздылығын атап өту маңызды, өйткені күрделілік қарама-қайшылықтардың жиналуына және біз көргеніміздей, қателіктердің пайда болуына әкеледі. Мен талғампаздықты минималды механизм мен максималды айқындықтың көмегімен берілген функционалдылыққа қол жеткізу ретінде анықтайтын едім".

Мұндағы негізгі идея - ең аз механизм. Басқаша айтқанда, әрбір функция, әрбір жүйелік қоңырау өз ауыртпалығын көтеруі керек. Дизайн тобының мүшесі жүйелік қоңырауды кеңейтуді немесе жаңа функцияны қосуды ұсынғанда, қалған әзірлеушілер одан: "егер олай етпесек, қорқынышты нәрсе бола ма?». Егер жауап: "жоқ, бірақ біреу бұл мүмкіндікті пайдалы деп санауы мүмкін" болса, оны амалдық жүйеге емес, пайдаланушы деңгейіндегі кітапханаға салыңыз, тіпті ол баяу болса да.

Әрбір функцияны оқтан жылдамырақ ету міндеті қойылмауы керек. Мақсат-Корбато механизмнің минимумы деп атаған нәрсені сақтау.

3-қағида: тиімділік

Үшінші Нұсқаулық-іске асырудың тиімділігі. Егер қандай да бір функцияны (немесе жүйелік қоңырауды) тиімді іске асыру мүмкін болмаса, оны жүзеге асыруға болмайды. Бағдарламашы әр жүйелік қоңырауды іске асыру және пайдалану құнын интуитивті түрде көрсетуі керек. Мысалы, UNIX жүйесіндегі бағдарламаларды жазатын бағдарламашылар lseek жүйелік қоңырауы read жүйелік қоңырауына қарағанда арзанырақ деп санайды, өйткені бірінші жүйелік қоңырау жадта сақталған меңзердің мазмұнын өзгертеді, ал екіншісі дискінің енгізу / шығару әрекетін орындайды. Егер интуиция бағдарламашыны сәтсіздікке ұшыратса, бағдарламашы тиімсіз бағдарламалар жасайды.,

Негізгі әдебиеттер: [1,2].
Бақылау сұрақтары:

1. PL/1 бағдарламалау тілі қай жылы жасалды?
2. С бағдарламалау тілі не үшін жасалған?
3. Әмбебап ОЖ үшін не маңызды? Тізім.
4. ОЖ үшін ең қиын міндет қандай?
5. Интерфейсті дамыту принциптерін сипаттаңыз.