Қазақстан Республикасының білім және ғылым министрлігі

Д. СЕРІКБАЕВ АТЫНДАҒЫ
ШЫҒЫС ҚАЗАҚСТАН МЕМЛЕКЕТТІК ТЕХНИКАЛЫҚ УНИВЕРСИТЕТІ

И.А. Котлярова, Л.Р. Сулейменова

ОПЕРАЦИЯЛЫҚ ЖҮЙЕЛЕР

5В070300 – «Ақпараттық жүйелер» және 5В070400 – «Есептеуіш техника және бағдарламалық қамтамасыз ету» бакалавриат мамандықтары үшін дәрістер кешені

Өскемен
Усть-Каменогорск
2013

ӘОЖ 004.451

И.А. Котлярова, Л.Р. Сулейменова Операциялық жүйелер. 5В070300 –«Ақпараттық жүйелер» және 5В070400 – «Есептеуіш техника және бағдарламалық қамтамасыз ету» бакалавриат мамандықтары үшін үшін дәрістер конспектісі. - Өскемен: Шығыс Полиграф, 2012, - 67 б.

Бұл дәрістер конспектісі «Операциялық жүйелер» пәнінен құралған. Университеттің оқу Кеңесінің шешімі бойынша бұл пән 5В070300 «Ақпараттық жүйелер», 5В070400 «Есептеуіш техника және бағдарламалық қамтамасыз ету» мамандықтарының оқу жоспарына енгізілген. Әрбір тақырып теориялық мәліметтерді, , өзін-өзі тексеруге арналған сұрақтарды қамтиды. Дәрістер кешені «Операциялық жүйелер» пәнінен операциялық жүйелер бойынша негізгі түсініктер, операциялық жүйелердің жіктелуі, түрлері, жадыны басқару, WINDOWS және LINUX операциялық жүйелері бойынша негізгі білімді береді. Бұл пән 5В070300 және 5В070400 мамандықтарының Мемлекеттік жалпы білім беру стандарттарымен қарастырылған техникалық профиль пәндер жиынтығына кіреді.

АТЭФ оқу-әдістемелік кеңесінің отырысында бекітілген

№____ хаттама ____ _____ 2013

МАЗМҰНЫ

1 ОПЕРАЦИЯЛЫҚ ЖҮЙЕЛЕРДІҢ. НЕГІЗГІ ТҮСІНІКТЕРІ. ОПЕРАЦИЯЛЫҚ ЖҮЙЕЛЕРДІҢ ЖІКТЕЛУ

1.1 Дәріс мақсаты

Дәріс мақсаты операциялық жүйелердің негізгі түсініктері, олардың қызметтері және операциялық жүйелердің жіктелуімен танысу.

1.2 Теориялық мәліметтер

1.2.1 Операциялық жүйе түсінігі. Операциялық жүйенің функциялары

Операциялық жүйе, ОЖ – компьютердің аппараттық бөлігі мен қолданбалы бағдарламаларын басқаратын сонымен қатар өзара және қолданушымен қатынасты қамтамасыз ететін бағдарламалық құралдардың жиынтығы.

Операциялық жүйе (ағылш. Operating system) — компьютердің аппараттық құралдарын басқаруды, файлдармен жұмыс жасауды, мәліметтерді енгізу және шығару, сонымен қатар қолданбалы бағдарламалар мен утилиттердің орындалуын қамтамасыз ететін комьютерлік бағдарламалардың базалық кешені.

Операциялық жүйе дербес компьютердің бағдарламалық қамтамасыздандыруының негізін құрайды. Операциялық жүйе қолданушының компьютермен өзара әрекеттесуін және барлық басқа бағдарламалардың орындалуын қамтамасыз ететін жүйелік және қызметтік бағдарламалық құралдардың кешені болып табылады.

Компьютерді қосқан уақытта операциялық жүйе жадыға басқа бағдарламалардан бұрын жүктелінеді де, одан кейін басқа бағдарламалар үшін олардың жұмыс істейтін платформасы мен ортасы қызметін атқарады.

Операциялық жүйе міндеттері:

- есептеу жүйесінде есептеу үдерістерін басқару;
- түрлі есептеу үдерістері арасында есептеу жүйесінің ресурстарын үлестіру;
- қолданушылардың қолданбалы бағдарламалары орындалатын бағдарламалық (операциялық) ортаны құру.

Операциялық орта – ОЖ функциялары мен қызметтерінің жиыны және оларды қолдану ережелері.

Операциялық орта – бағдарламалар мен қолданушыларға белгілі бір қызметтерді қолдану мақсатында ОЖ қатынауға қажетті интерфейстер жиыны.

Жалпы алғанда ОЖ бірнеше операциялық орта болуы мүмкін. Операциялық ортада бірнеше интерфейс болуы мүмкін: қолданушы интерфейстері және бағдарламалық интерфейстер.

Операциялық орта – осы ортаның жұмыс істеу ережесіне сәйкес құрылған бағдарламалар орындалуы мүмкін жүйелік бағдарламалық аймақ.

Операциялық жүйе келесі функцияларды орындайды:

- дербес компьютердің әрбір блогының жұмысын және олардың өзара әрекеттесуін басқару;
- сыртқы жадыда ақпаратты сақтауды ұйымдастыру;
- қолданушы интерфейсін ұсыну (қолданушыдан командалық жол түрінде немесе манипулятордың (тінтуірдің) көмегімен берілген тапсырмалармен командаларды қабылдау);
- бағдарламаларды (қосымшаларды) жедел жадыға жүктеу және орындау;
- сыртқы құрылғыларға стандартталған қатынау (енгізу/шығару операцияларын ұйымдастыру және басқару);
- жедел жадыны басқару (үдерістер арасында үлестіру, виртуалды жадыны ұйымдастыру);

- мультибағдарламалау режимін қамтамасыз ету, екі және одан да көп тапсырмаларды бір процессорда орындау;
- нақты уақыт жүйелерінде жауап берудің минималды уақытын қамтамасыз ету;
- тапсырмаларды жоспарлау және диспетчеризациялау;
- жүйелік ресурстарды, мәліметтермен қолданушылардың бағдарламаларын, орындалып жатқан үдерістерді және өзін қолданушылармен олардың бағдарламаларының қасақана әрекеттерінен қорғау, жүйе жартылай жаңылысқан жағдайда қызмет көрсету;
- аутентификация (қолданушы шынында да өзі ме, жоқ па екендігін тексеру), авторизация (өзін ұсынған қолданушының қандайда бір операцияны орындауға құқығы бар ма, жоқ па екендігін тексеру) және қауіпсіздікті қамтамасыз етудің басқа да құралдары.

Жалпы алғанда, операциялық жүйе компьютердің сыртқы жадысы – дискіде сақталынады. Компьютерді желіге қосқаннан кейін операциялық жүйені дискіден жедел жадыға жазу үдерісі басталады. Бұл үдеріс операциялық жүйені жүктеу деген атау алды.

Дербес компьютерді қосқаннан кейін оның процессоры жұмысын бастайды.

Бірінші орындалатын команда BIOS адрестік кеңістігіне тиесілі. Бұл команда жай ғана басқаруды BIOS инициялизациялау бағдарламасына береді.

BIOS инициялизациялау бағдарламасы POST бағдарламасының көмегімен компьютер құрылғыларының дұрыс жұмыс істейтіндігін тексереді және оларды инициализациялайды.

Одан кейін BIOS алдын ала құрылған тізімде көрсетілген құрылғылардан жүктеу құрылғысын тапқанға дейін сұрай береді. Егер бұл сияқты құрылғы табылмаса, онда қате туралы хабарлама шығарылып, жүктеу барысы тоқтатылады. Егер BIOS жүктеу құрылғысын тапса, онда ол одан бастапқы жүктеуішті оқып, оған басқаруды береді.

Қатті диск жағдайында бастапқы жүктеуіш басты жүктеуіш жазба (MBR) деп аталады және көбінесе операциялық жүйеге тәуелді емес болады. Жалпы ол қатты дискінің белсенді бөлімін іздейді, бұл бөлімнің жүктеуіш секторын жүктеп, басқаруды соған береді. Жүктеуіш сектор жадыға операциялық жүйе ядросын жүктеп, оған басқаруды беруі керек.

Операциялық жүйе жүктеуіші – тура компьютер қосылғаннан кейін операциялық жүйенің жүктелуін қамтамасыз ететін жүйелік бағдарламалық қамтамасыздандыру.

Бастапқы жүктеуіш қатты дискіде, дискетада, CD-ROM-да орналасуы мүмкін және тіптен желілік тақшаның көмегімен де алынады. Сондықтанда компьютер белгілі бір ретпен аталған құрылғылардан сұрау жүргізеді, компьютер сұрауды қажетті ақпаратты тапқанға дейін жүргізе береді (ОЖ жүктеуді жүргізуге болатын құрылғылардың ретін BIOS-та баптауға болады).

Сыртқы жүйелік құрылғы – ОЖ жүктеу жүргізілетін құрылғы – дискті қозғағыш, қатты диск, CD-ROM, флеш-диск, желілік тақша.

1.2.2 Операциялық жүйелердің жіктелінуі

Операциялық жүйелер компьютер ресурстарын (процессорлар, жады, құрылғыларды) басқару алгоритмдерін жүзеге асыру ерекшеліктері, қолданылған жобалау тәсілдерінің ерекшеліктері, аппараттық платформалардың типтері, қолдану аймағы және тағы басқа көптеген қасиеттері бойынша өзгешеленеді.

ОЖ кейбір негізгі белгілері бойынша жіктелінуін қарастырайық.

Процессорларды басқару алгоритмдеріне қарай ОЖ келесі түрлерге бөлінеді:

- бір тапсырмалы және көп тапсырмалы;
- бір қолданушылы және көп қолданушылы;
- бір процессорлы және көп процессорлы;
- жергілікті және желілік.

Бір мезетте орындалатын тапсырмалар саны бойынша ОЖ екі класқа бөлінеді:

- бір тапсырмалы (MS DOS, MSX);
- көп тапсырмалы (OS/2, Unix, Windows).

Бір тапсырмалы жүйелерде сыртқы құрылғыларды басқару құралдары, файларды басқару құралдары, қолданушылармен байланысу құралдары қолданылады. Көп тапсырмалы ОЖ бір тапсырмалы жүйелерге тән құрылғылардың барлығын пайдаланады, сонымен қатар бірлесе қолданылатын ресурстарды бөлуді басқарады. Бірлесе қолданылатын ресурстар: процессор, жедел есте сақтау құрылғысы (ОЕСҚ), файлдар және сыртқы құрылғылар.

Сонымен қатар, кез келген көп тапсырмалы жүйе көп қолданушылы, ал кез келген бір қолданушылы жүйе бір тапсырмалы бола бермейтінін атап өткен жөн.

Ығыстырылатын және ығыстырылмайтын көп тапсырмалылық. Процессор уақыты маңызды бөлінісілетін ресурс болып табылады. Жүйеде бір мезетте болатын үдерістер (немесе тармақтар) арасында процессор уақытын бөлу тәсілі көбінесе ОЖ ерекшеліктерін анықтайды. Көп тапсырмалылықты жүзеге асырудың қазіргі уақытта бар көптеген нұсқаларының ішінен алгоритмдердің екі тобын ерекшелеп көрсетуге болады:

- ығыстырылмайтын көп тапсырмалылық (NetWare, Windows 3.x);
- ығыстырылатын көп тапсырмалылық (Windows NT, OS/2, UNIX).

Көп тапсырмалылықтың бұл екі нұсқасының арасындағы басты айырмашылығы үдерістерді жоспарлау механизмінің орталықтандырылу дәрежесінде. Бірінші жағдайда үдерістерді жоспарлау механизмі тұтасымен операциялық жүйенің назарына аударылған, ал екінші жағдайда жүйемен қолданбалы бағдарламалар арасында үлестірілген. Ығыстырылмайтын көп тапсырмалылықта белсенді үдеріс басқаруды операциялық жүйеге, ол кезектен орындалуға дайын басқа үдерісті орындау үшін таңдап алғанға өз еркімен бергенге дейін орындала береді. Ығыстырылатын көп тапсырмалылықта процессорды бір үдерістен келесі үдеріске аудару туралы шешімді белсенді үдеріс емес, операциялық жүйе қабылдайды.

Көп тармақтылықты қолдау. ОЖ маңызды қасиеттерінің бірі бір тапсырма көлемінде есептеулерді параллельдеу мүмкіндігі болып табылады. Көп тармақты ОЖ процессор уақытын тапсырмалар арасында емес, тапсырмалардың жекелеген тармақтары (тарамдары) арасында бөледі.

Қолдану аймағына қарай көп тапсырмалы ОЖ үш типке бөлінеді:

- пакеттік өңдеу жүйелері (ЕС ОЖ);
- уақытты бөлісуі бар жүйелер (Unix, Linux, Windows);
- нақты уақыт жүйелері (RT11, QNX).

Пакеттік өңдеу жүйелері негізінен жылдам нәтиже алуды қажет етпейтін есептеулер жүргізу сипатындағы тапсырмаларды шешуге арналған. Пакеттік өңдеу ОЖ басты мақсаты максималды өткізгіштік қабілет немесе уақыттың бір бірлігінде тапсырмалардың максималды санын шешу болып табылады. Бұл мақсатқа қол жеткізу үшін пакеттік өңдеу жүйелерінде келесі қызмет ету сұлбасы пайдаланылады: жұмыс басында тапсырмалар пакеті құрылады, әрбір тапсырмада ресурстарға қойылатын талаптар берілген; бұл тапсырмалар пакетінен мультибағдарламалық қосынды, яғни бір мезетте орындалатын тапсырмалар жиыны құрылады. Есептеу машинасының барлық құрылғыларына жүктеме бірдей болуы үшін бір мезетте орындау үшін ресурстарға түрлі талаптар қоятын тапсырмалар таңдап алынады. Бұл сияқты ОЖ қандай да бір тапсырманың нақты уақыт аралығында орындалатындығына кепілдеме беру мүмкін емес. Пакеттік өңдеу жүйелерінде бір тапсырманы орындаудан келесі тапсырманы орындауға көшу тек белсенді тапсырма өзі процессордан бас тартқан жағдайда ғана орындалады, мысалы, енгізу-шығару операциясын орындаудың қажеттілігінен. Сондықтан да бір тапсырма процессорды ұзақ уақыт босатпауы мүмкін, ал бұл интерактивті тапсырмаларды орындауды мүмкін етпейді.

Бұл жүйелер үлкен көлемді ақпараттарды өңдеуде жоғары өнімділікті қамтамасыз етеді, бірақ қолданушының интерактивті режимдегі жұмысының тиімділігін төмендетеді.

Уақытты бөлісуі бар жүйелерде әрбір тапсырманы орындау үшін шағын уақыт аралығы бөлінеді, және ешбір тапсырма процессорды ұзақ уақытқа алмайды. Егерде бұл уақыт аралығы минималды болып алынған болса, онда бірнеше тапсырма бір мезетте орындалғандай болып көрінеді. Бұл жүйелердің өткізгіштік қабілеті төмен, бірақ қолданушының интерактивті режимдегі жұмысының жоғары тиімділігін қамтамасыз етеді.

Нақты уақыт жүйелері технологиялық үдерістерді немесе техникалық объектілерді басқару үшін қолданылады, мысалы ұшу объектісі, станок және т.б.

Аталған жағдайлардың барлығында объектіні басқаратын қандайда бір бағдарлама орындалуға тиіс шекті мүмкін уақыт мөлшері бар, олай болмағанда апат болуы мүмкін: серіктің көріну аумағынан шығып кетуі, тетіктерден түсіп тұрған сараптаматық мәліметтердің жоғалуы мүмкін. Сонымен, нақты уақыт жүйелерінің тиімділік белгісі бағдарламаны іске қосу мен нәтижені (басқарушы ықпалды) алу арасында алдын ала берілген уақыт интервалын сақтай алу қабілеті болып табылады. Бұл уақыт жүйе реакциясының уақыты деп аталады, ал жүйенің сәйкес қасиеті- реактивтілік деп аталады. Бұл жүйелер үшін мультибағдарламалық қосынды аладын ала даярланған бағдарламалардың тиянақталған жиыны ретінде қабылданады, ал орындауға бағдарламаны таңдау объектінің ағымды күйін ескере отырып немесе жоспарлық жұмыстардың кестесіне сәйкес таңдалынады.

Кейбір операциялық жүйелерде әртүрлі типті жүйелердің қасиеттері болуы мүмкін, мысалы, тапсырмалардың бір бөлігі пакеттік өңдеу режимінде, ал енді бір бөлігі – уақытты бөлісуі бар режимінде немесе нақты уақыт режимінде орындалуы мүмкін. Бұл сияқты жағдайда көбінесе пакеттік өңдеу режимін фондық режим деп атайды.

ЭЕМ бір мезетте жұмыс істейтін қолданушылар саны бойынша ОЖ бір қолданушылы (MS DOS, Windows 3.x, OS/2 ертеректегі нұсқалары) және көп қолданушылы (Unix, Linux, Windows 95 – XP) болып бөлінеді.

Көп қолданушылы ОЖ әрбір қолданушы өзі үшін қолданушы интерфейсін баптай алады, яғни ол өзінің жазба белгі жиындарын, бағдарламалар тобын құрып, жеке түстік сұлбаны көрсете алады, тапсырмалар панелін өзіне ыңғайлы орынға көшіріп, «Пуск» мәзіріне жаңа пункттілер қоса алады.

Көп қолданушылы ОЖ әрбір қолданушының ақпараттарын басқа қолданушылардың рұқсатсыз қатынауынан қорғау құралдары бар.

Бір процессорлы және көп процессорлы операциялық жүйелер. ОЖ тағы бір маңызды қасиеттерінің бірі онда көп процессорлы өңдеуді қолдау құралдары – мультипроцессорлеудің болуы немесе болмауы болып табылады. Мультипроцессорлеу ресурстарды басқару алгоритмдерін күрделендіруге әкеледі.

Қазіргі таңда ОЖ мәліметтерді көп процессорлы өңдеу функцияларын енгізу жаппай етек алуда. Бұл сияқты функциялар Sun фирмасының Solaris 2.x, Santa Crus Operations компаниясының Open Server 3.x, IBM фирмасының OS/2, Microsoft фирмасының Windows NT және Novell фирмасының NetWare 4.1 операциялық жүйелерінде бар.

Есептеу үдерісін ұйымдастыру тәсілі бойынша бұл ОЖ асимметриялық және симметриялық болып бөлінуі мүмкін.

Асимметриялық ОЖ қолданбалы тапсырмаларды басқа процессорлар арасында үлестіре отырып, жүйе процессорларынының тек бірінде ғана орындалады. Симметриялық ОЖ толығымен орталықсыздандырылған, сонымен қатар жүйелік және қолданбалы тапсырмалар арасында процессордарды бөле отырып, процессорлардың барлық мүмкіндігін пайдаланады.

ЭЕМ жіктеудің тағы бір маңызды білгілерінің бірі оларды жергілікті және желілікке бөлу.

Жергілікті ОЖ автономды ДК немесе компьютерлік желілерде клиент ретінде қолданылатын ДК пайдаланылады. Жергілікті ОЖ құрамына қашықтатылған ресурстармен қызметтерге қатынауға арналған бағдарламалық қамтамасыздандырудың клиенттік бөлігі кіреді.

Желілік ОЖ ресурстарын ортақ пайдалану мақсатында желіге қосылған ДК ресурстарын басқаруға арналған. Олар ақпаратқа қатынауды, оның біртұтастығын шектеудің маңызды құралдарын және де желілік ресурстарды пайдаланудың басқа мүмкіншіліктерін ұсынады. Желілік ОЖ құрамында автономды ОЖ мүлдем қажет емес байланыс желілері арқылы компьютерлер арасында хабарламалар жіберу құралдары бар. Осы хабарламалардың негізінде желілік ОЖ желіге қосылған қашықтатылған компьютерлер арасында компьютер ресурстарын бөлуді қамтамасыз етеді. Хабарламаларды жіберу функциясын орындау үшін ОЖ арнайы бағдарламалық компоненттер бар. Олар IP, IPX, Ethernet және т.б. сияқты танымал коммуникациялық хаттамаларын жүзеге асырады.

Операциялық жүйенің қасиеттеріне ол өзі бағдарланған аппараттық құралдар тікелей әсер етеді. Аппаратураның типі бойынша операциялық жүйенің келесі түрлерін ажыратады: дербес компьютердің ОЖ, мини-компьютердің ОЖ, мейнфреймдердің ОЖ, кластерлермен ЭЕМ желілерінің ОЖ. Аталған компьютерлер тізімі арасында бір процессорлысы да, көп процессорлысы да кездесуі мүмкін. Қандай жағдайда болмасын аппараттық құралдардың ерекшелігі операциялық жүйенің ерекшеліктеріне әсер етеді.

Кластерлердің операциялық жүйесіне басқа талаптар қойылады. Кластер – ортақ бағдарламалардың орындалуы үшін бірлесе жұмыс істейтін және қолданушыға бір жүйе түрінде көрінетін бірнеше есептеу жүйелерінің әлсіз байланысқан жиынтығы. Кластерлік жүйелердің қызмет етуі үшін қажет арнайы аппаратурамен қатар, операциялық жүйе тарапынан бағдарламалық қолдау қажет. Ал бұл бағдарламалық қолдау бөлінісілетін ресурстарға қатынауды синхронизациялау, жүйенің бас тартуларын және динамикалық реконфигурациясын табуды іске асырады. Кластерлік технологиялар саласындағы ең алғашқы шешімдердің бірі VAX компьютерлері негізіндегі Digital Equipment компаниясының шешімі болды. Жуырда бұл компания Microsoft корпорациясымен Windows NT қолданатын кластерлік технологияны әзірлеу туралы келісімге келді. Кейбір компаниялар UNIX-машиналары негізіндегі кластерлерді ұсынуда.

Аппараттық платформаның нақты бір типіне бағдарланған ОЖ қатар, бір компьютерден екінші компьютерге оңай көшіріле алатын арнайы құрылған ОЖ бар, оларды мобильді ОЖ деп атайды. Қазіргі уақытта бұл сияқты ОЖ мысал ретінде танымал UNIX ОЖ келтіруге болады. Бұл сияқты жүйелерде аппараттық-тәуелді орындары мейлінше оқшауланған, сондықтан жүйені жаңа платформаға көшіргенде тек олар көшіріледі. ОЖ басқа бөліктерін көшіруді жеңілдету құралы, оның кодын машиналық-тәуелсіз тілде жазу, мысалы, операциялық жүйелерді бағдарламалауға арнап шығарылған бағдарламалау тілі С.

1.3 Бақылау сұрақтары

1.3.1 ОЖ түсінігі. ОЖ тағайындаулары.
1.3.2 ОЖ қызметтері?
1.3.3 ОЖ жіктелуі қандай?
1.3.4 Пакеттік өңдеу жүйелеріне қандай ерекшеліктер тән?
1.3.5 Уақытты бөлісуі бар жүйелерде қандай ерекшеліктер тән?
1.3.6 Нақты уақыт жүйелеріне қандай ерекшеліктер тән?

2 ОПЕРАЦИЯЛЫҚ ЖҮЙЕЛЕРДІҢ ҚҰРЫЛЫМЫ МЕН ТҰРҒЫЗЫЛУ ПРИНЦИПТЕРІ

2.1 Дәріс мақсаты

Дәріс мақсаты операциялық жүйелердің құрылымы мен тұрғызылу принциптерімен танысу.

2.2 Теориялық мәліметтер

2.2.1 ОЖ тұрғызу әдістерінің ерекшеліктері

Операциялық жүйені сипаттағанда көбінесе олардың негізіне салынған құрылымдық ұйымдастырылуының ерекшеліктерін және негізгі тұжырымдамасын көрсетеді. Бұл сияқты негізгі тұжырымдамалар:

– Жүйе ядросының тұрғызылу тәсілдері– монолитті ядро немесе микроядролық тұрғыдан қарау. ОЖ көбінде монолитті ядроны пайданылады, ол басым режимде жұмыс жасайтын, бір бағдарлама ретінде құрастылылады және бір процедурадан келесісіне жылдам өтуді пайдаланып, басым режимнен қолданушы режиміне өтуді және керісінше істеуді талап етпейді. ОЖ микроядро негізінде тұрғызу балама болып табылады. Ол да басым режимде жұмыс жасап, тек аппаратураны басқару бойынша минимум функцияларды атқарып, ал сол мезетте ОЖ жоғарырақ деңгейдегі функцияларын ОЖ мамандандырылған компоненттері – қолданушы режиминде жұмыс жасайтын серверлер орындайды. Бұл сияқты тұрғызылғанда ОЖ баяуырақ жұмыс істейді, өйткені басым режиммен қолданушы режиміне өту және керісінше өтулер жиі орындалады, бірақ жүйе иілгіштеу болады, себебі қолданушы режимінің серверлерін қосу, жою, өзгерту арқылы оның функцияларын өсіруге, өзгертуге және азайтуға болады. Сонымен қатар, кез келген қолданушы үдерістері сияқты серверлер бір бірінен жақсы қорғалған. Микроядерлі ОЖ - QCPB QNX, ал монолитті ОЖ - Windows 9x және Linux жатады. Windows 9x ОЖ үшін қолданушы ядроны өзгерте алмайды, себебі оның қолынды ядроны жинақтаудың бастапқы коды және бағдарламасы жоқ. Ал Linux ОЖ бұл сияқты мүмкіндік берілген, яғни қолданушы қажет бағдарламалық модульдерді және драйверлі қосып, ядроны өзі жинақтай алады.

– ОЖ объектілі-бағдарланған тұрғы негізінде тұрғызу оның операциялық жүйе ішінде қосымшалар деңгейінде өздерін жақсы көрсете білген барлық құндылықтарын қолдануға мүмкіндік береді. Атап айтсақ, сәтті шешімдерді стандартты объект формасында аккумуляциялау, мұрагерлік механизмінің көмегімен бар объектілердің негізінде жаңа объектілер құру мүмкіндігі, объектілердің ішкі құрылымдарына инкапсуляциялау арқылы мәліметтерді жақсы қорғау, ал бұл өз кезегінде мәліметтерді сырттан рұқсатсыз пайдалануға мүмкіндік бермейді, жақсы анықталған объектілер жиынынан тұратын жүйенің құрылымданғандылығы.

– Бірнеше қолданбалы ортаның бар болуы бірнеше ОЖ үшін жасалған қосымшаларды бір ОЖ шеңберінде орындауға мүмкіндік береді. Көптеген заманауи операциялық жүйелер MS-DOS, Windows, UNIX (POSIX), OS/2 қолданбалы орталарын бір мезетте қолдайды. Көптік қолданбалы орта тұжырымдамасы микроядро негізіндегі ОЖ іске асырылады, бұл микроядро негізіндегі операциялық жүйелермен түрлі серверлер жұмыс жасайды, ал олардың біразы қандайда бір операциялық жүйенің қолданбалы ортасын жүзеге асырады.

– Операциялық жүйенің үлестірілген түрде ұйымдастырылуы қолданушылар мен бағдарламалаушылардың желілік ортадағы жұмысын жеңілдетеді. Үлестірілген ОЖ қолданушыларға желіні дәстүрлі бір процессорлы компьютер сияқты елестетуге және қабылдауға мүмкіндік беретін механизмдер бар. ОЖ үлестірілген түрде ұйымдастырудың өзіндік белгілері бар: бөлінісілетін ресурстардың біртұтас анықтамалық қызметінің бар

болуы, уақыттың біртұтас қызметі, бағдарламалық процедураларды машиналарға тұнық үлестіруге арналған жойылған процедараларды шақыру (RPC) механизмін қолдану, бір тапсырма көлемінде есептеулерді параллельдеп, және оларды желінің бірнеше компьютерінде орындауға мүмкіндік беретін төп тармақты өңдеу, және тағы басқа үлестірілген қызметтер.

2.2.2 Монолитті ядро және микроядро

ОЖ модульдері реентерабельді емес. Қарапайым ОЖ жоспарлаушыдан басқа көптеген модульдер кіреді – енгізу/шығарудың ішкі жүйесі, файлдық жүйе немесе жүйелер, жады диспетчері және т.б. Іс жүзінде осы барлық модульдер бөлінісілетін ресурстармен жұмыс жасайды және реентерабельді бола алмайды.

Монолитті ядро. Бұл мәселелердің ең бірінші шешімдерінің бірі ОЖ реентерабельді емес модульдерінің жұмысы барысында тапсырмалардың ауыстырылуына тыйым салу болды. Бұл уақытта ОЖ барлық модульдері ядро (kernel) деп аталатын конгломератқа жинақталады. Ядроны басым тапсырма деп атасақ болады, бірақ ол толық мағынасында үдеріс болып табылмайды, себебі жоспарлаушы ядродан процессорды тартып ала алмайды. Ядрода барлық реентерабельді емес модульдер жинақталған, көп жағдайларда реентерабельді модульдерде жинақталады. Жоспарлаушының өзі де ядроның бөлігі болып табылады.

Жүйелік шақыруды орындай отырып, қолданушы бағдарламасы басқаруды ядроға береді. Ядроға кірген тұста үдерістерді жоспарлау тоқтатылады. Ядро немесе сұратуды орындайды, не болмаса оны орындауға кезекке қойып, басқаруды жоспарлаушыға береді. Жоспарлаушы ағымдағы үдерісті (егер ол белсенді болса) тізімнің соңына қойып, интервал ретінде уақыт квантын беріп, таймерді бағдарламалайды, және басқаруды келесі белсенді үдеріске береді.

Үдеріс басқаруды екі түрлі тәсілмен жоғалтып алуы мүмкін – уақыт кванты біткен мезетте немесе жүйелік шақыруды орындау барысында. Бірінші жағдайда ол үнемі белсенділер кезегінде қалады, ал екінші жағдайда, жасалынған шақыртуға байланысты ол қалуы да, қалмауы да мүмкін. Енгізу/шығару және синхронизация сұратулары басқа үдерістермен бірге көбінде үдерістің күту күйіне ауысуына әкеледі. Ядро басқаруды тек аппараттық үзілімдер кезінде ғана жоғалта алады. Үзілімдер арнайы бағдарламалар – құрылғылар драйверлерімен өңделеді. Драйверлер ядроның бөлшегі болып табылады және олардың қарапайым жүйелік шақыруларды орындауға құқықтары жоқ. Ядроның қандайда бір модулі белсенді болса, ешқандай қолданушы үдерісі басқаруды ала алмайды. Бұл сияқты архитектура монолитті ядро деп аталады.

Монолитті ядроның басты кемшілігі нақты уақыт жүйелерін жүзеге асыруда туындайтын қиыншылықтар болып табылады. РВ үдерістері кепілдемелі уақыт интервалында басқаруды алулары қажет. Мүмкіндігінше бұл интервал үлкен болмауы керек. ОЖ кейбір модульдерінің орындалу уақыты үлкен болуы мүмкін, мысалы директориядан файлды іздеу. Сондықтан, егер біз нақты уақыт жүйелерін жақсы реакция уақытымен жүзеге асырғымыз келсе, онда біз РВ үдерістеріне ядроға қарағанда жоғарырақ басымдылық беруіміз керек. Бірақ, егер үдерістің басымдылығы ядродан жоғары болса, онда ол жүйенің реентерабельді емес модульдерінің жұмыс істеген уақытында басқаруды өзіне алуы мүмкін.

Микроядро. Бұл қиыншылықтың шешімі болып ОЖ реентерабельді емес модульдері біртұтас (бүтін) сыни ресурс ретінде қарастырылатын архитектура болып табылады. Реентерабельді емес модульдерге жүгіну, мысалы енгізу/шығарудың ішкі жүйесіне жүгіну шақыру ретінде емес, ал сұратуды кезекке қою түрінде жүзеге асады, яғни сұратуды өңдеу бірден жүргізілмейді, ал кішкене уақыттан кейін жүзеге асуы мүмкін. Сондықтан біз іс жүзінде сұратуды кез келген уақытта кезекке қоя аламыз, тіптен сұратуларды өңдеушінің жұмыс істеп тұрған уақытында да.

Біз сұратуды дәл осы кезектің үстінен басқа операциялар жүріп жатқанда қоя алмаймыз, бірақ тізім(нен/ге) элементті шығару/қою уақыты үш-төрт команданың орындалу уақытына тең және бұл уақытқа үзілімдерге тыйым салуға болады.

Бұл ойдың дамуы 80-жылдарда микроядро (microkernel) атымен белгілі архитектураның дамуына әкелді. Микроядро – жоспарлаушыдан және синхранизация үдерістері арасында мәліметтерді таратудың базалық құралдарынан тұратын операциялық жүйенің минималды функционалды толық бөлігі.

Сонымен, микроядро басқа жүйелік қызметтермен қосымшаларға сүйенетін операциялық жүйенің негізгі функцияларын жүзеге асырады. Микроядролық ОЖ құрастырудағы басты қиыншылық ядродан шығарылуы мүмкін функцияларды танып білу болып табылады. ОЖ файлдық жүйе, терезелерді басқару жүйесі және қауіпсіздік қызметі сияқты компоненттері шеткері (сыртқы) модульдерге айналады.

Микроядроға қосылмаған жүйе модульдері ядромен де және бір-бірімен өзара да әрекеттесетін жеке үдерістер болып табылады. Егер қолданушының үдерісіне файлды ашу керек болса, онда ол сәйкес жүйелік үдеріске сұрату жібереді де жауап күтеді.

Бұл сияқты архитектура жүйе ядросының реентерабельділігіне байланысты барлық қиыншылықты шешіп және нақты уақыт үдерістерінеде және тіпті үзілім өңдеушілерінеде жүйелік шақыруларды еш шектеусіз орындауға мүмкіндік береді. Мұның соңғысы басқа драйверлерге немесе жүйе ядросына қатынайтын драйверлерді жазуға болатындығын білдіреді.

Архитектуралардың өзгешелігі. Монолитті және микроядроның өзгешелігі микроядро негізіндегі жүйедегі жүйелік және қолданушылар үдерістері бірдей жоспарланып және мәліметтер алмасумен синхронизация үшін бір жүйелік шақыртуларды пайданалады. Ал монолитті ОЖ ядро үдерістерімен қолданушылар үдерістері әртүрлі жсопарланып, және әркеттесу үшін көбінесе әртүрлі құралдарды пайдаланады. Сонымен, микроядролық архитектураларда операциялық жүйенің функцияларын тігінен үлестіру көлденеңмен алмастырылды. Микроядродан жоғары тұрған компоненттер хабарламалармен алмасу үшін микроядро құралдарын пайдаланады, бірақ өзара тікелей әрекеттеседі.

Басымдылықтары мен кемшіліктері. Микроядролық ОЖ модульдері жақсы құрылымдалған – жады диспетчері, файлдық жүйелер, құрылғылар драйверлері және т.б. – бір бірімен айрықшаланған нүктеде өзара әрекеттесуі керек және құрылымдық және модульдік бағдарламалаудың тұжырымдамасына толық сәйкес келетін жақсы анықталған интерфейсті қолдануы керек. Тағы бір айтатыны, микроядролық жүйелердің бұл қасиеті оларды үлестірілген ортада қолдануға мүмкіндік береді.

Микроядроның тағы бір басымдылығы оның тасымалдануы (өйткені ОЖ барлық машиналық-тәуелді бөлігі микроядрода оңашаланған, жүйені жаңа процессорға көшіру үшін көп өзгерістер қажет емес және бұл өзгерістер логикалық түрде топтастырылған) және масштабталуы (микроядролық ұйымдастыру микроядроның интерфейсінің шектеулі жиынына сүйенетін кеңейтуді қолдайды).

Микроядролық архитектураның кемшіліктеріне хабарламалармен алмасу жүйелік функцияларды шақыртудан баяурақ жүретіндігін жатқызуға болады. Бұл құрастырушылардан микроядролық ОЖ модульдері алмасатын хабарламалардың құрылымын тиімдеу шараларын қарқынды жүргізуін талап етеді.

2.2.3 Басым режимдегі ядро

Қосымшалардың жұмыс барысын сенімді басқару үшін операциялық жүйе оларға қарағанда қандайда бір басымдылықтарға ие болуы керек. Олай болмаса, бұрыс жұмыс жасап тұрған қосымша ОЖ жұмысына араласуы мүмкін, мысалы, оның кодтарының бөліктерін бүлдіруі мүмкін. Егер операциялық жүйені құрастырушылардың шешімдері қосымшалардан қорғалмаған жүйе модульдеріне орналастырылса, бұл шешімдердің қаншалықты элегентты және тиімді болғанына қарамастан-ақ, олардың барлық әрекеттері

нәтижесіз болады. Операциялық жүйенің ерекше өкілеттілігі болуы керек, себебі мультибағдарламалық режимде қосымшалардың компьютер ресурстарына таласы болған жағдайда арбитр ролін ойнау үшін керек. Ешбір қосымшаның ОЖ бақылауынсыз қосымша жады аймағын алу, ОЖ рұқсат етілген уақыт аралығынан артық процессорды ұстау, ортақ қолданылатын сыртқы құрылғыларды тікелей басқаруға мүмкіндіктері болмайды.

Арнайы аппараттық құрылғылардың көмегінсіз операциялық жүйеге басымдылықты беру мүмкін емес. Компьютер аппаратурасы жок дегенде екі жұмыс режимінде жұмыс жасауы керек – қолданушы режимі (user mode) және басым режимде, сонымен қатар, ядро режимі (kernel mode), немесе супервизор режимі (supervisor mode) деп те атайды. Яғни операциялық жүйе немесе оның кейбір бөлімдері басым режимде, ал қосымшалар – қолданушы режимінде жұмыс жасайды.

Ядро ОЖ барлық функцияларын орындайды, көбінде ядро ОЖ басым режимде жұмыс істейтін бөліміне айналады. Кейде бұл қасиет – басым режимдегі жұмыс – «ядро» түсінігінің негізгі анықтамасы болады.

Қосымшалар бағыныңқы күйге қойылады. Бұл қолданушы режимінде процессорды тапсырмадан тапсырмаға ауыстыру, енгізу-шығару құрылғыларын басқару, үлестіру және жадыны қорғау механизмдеріне қол жеткізумен байланысты кейбір сыни командаларды орындауға тыйым салу арқылы жүргізіледі. Қолданушы режимінде кейбір нұсқауларды орындауға шартсыз тыйым салынады (бұл сияқты нұсқауға басым режимге өту нұсқаулығы жататыны анық), ал кейбіреулерін орындауға белгілі бір жағдайларда ғана тыйым салынады. Мысалы, қатты дисктің контроллеріне қатынауда барысында енгізу-шығару нұсқаулықтары қосымшаларға тыйым салынған болуы мүмкін, қатты диск ОЖ және барлық қосымшаларға ортақ мәліметтерді сақтайды, бірақ нақты бір қосымшаның монопольді иелігіне бөлінген тізбектелген портқа қатынауда рұқсат етілген. Маңыздысы сыни нұсқаулықтарды орындауға рұқсат беретін шарттар толығымен ОЖ бақылауында болады және бұл бақылау қолданушы режимі үшін шарттсыз тыйым салынған нұсқаулықтар жиыны арқасында жүзеге асырылады.

ОЖ жадыға қатынау басымдылығыда дәл осы сияқты жүргізіледі. Мысалы, жадыға қатынау нұсқаулығын орындау егер нұсқау ОЖ бұл қосымшаға бөлінген жады аумағына қатынаса қосымшаға рұқсат етіледі, ал егер ОЖ немесе басқа қосымшалар тұрған жады аумағына қатынаса, рұқсат етілмейді. ОЖ жадыға қатынауды толық бақылауы жадыны қорғау механизмдерін конфигурациялау нұсқаулығы немесе нұсқаулықтарын (мысалы, IBM мэйнфрейдарында жадыны қорғау кілттерін немесе Pentium процессорларының жадысындағы дескрипторлар кестесі көрсеткішін өзгерту) тек басым режимде ғана орындауға болатындығының арқасында мүмкін болады.

Операциялық жүйе жадыны қорғау механизмін тек өз жады аумағын қосымшалардан қорғау үщін ғана емес, сонымен бірге ОЖ қандайда бір қосымшаға бөлінген жады аумағын басқа қосымшалардан қорғау үшін де қолданады. Әрбір қосымша өзінің адрестік кеңестігінде жұмыс істейді деп айтады. Бұл қасиеті бұрыс жұмыс жасап тұрған қосымшаларды өз жады аумағына оқшаулайды, содан оның қателіктері басқа қосымшаларға және ОЖ әсер етпейді.

Аппаратты жүзеге асырылатын басымдылық деңгейлерінің саны, және ОЖ құптайтын басымдылық деңгейлерінің саны арасында тура сәйкестік жоқ. Intel компаниясының төрт деңгейді қамтамасыз ететін процессорлары негізінде OS/2 операциялық жүйесі үш деңгейлі басымдылық жүйесін, ал Windows NT, UNIX және тағы басқа кейбір операциялық жүйелер екі деңгейлі жүйемен шектелуде.

Ал, бір жағынан егер аппаратурада тым болмағанда басымдылықтың екі деңгейі болса, онда ОЖ осының негізінде бағдарламалық жолмен жетік қорғаныс жүйелерін жасайды.

Бұл жүйе, мысалы иерархияны құрап тұрған басымдылықтардың бірнеше деңгейін құптай алады. Басымдылықтың бірнеше деңгейінің болуы ОЖ модульдерінің арасында

өкілеттіліктерді дәлірек бөлуді де және қосымшалар арасында да дәлірек бөлуді мүмкін етеді. ОЖ ішінде басымдырақ өкілеттілікті бөліктермен қатар басымдылығы төмен өкілеттілікті бөліктердің болуы бағдарламалық кодтардың ішкі қателіктеріне ОЖ бекемділігін арттырады. Себебі бұл қателіктер белгілі бір деңгейлі басымдылықты модульдердің ішіне ғана тарайды. Қолданбалы модульдер ортасында басымдылықтарды дифференциациялау күрделі қолданбалы кешендерді тұрғызуға мүмкіндік береді, мұнда басымдылығы жоғарырақ модульдер, мысалы басымдылығы төменірек модульдердің мәліметтеріне қатынап және олардың орындалуын басқара алады.

ОЖ процессордың басымдылықтарының екі режимінің негізінде ресурстарды жекелей қорғаудың күрделі жүйесін тұрғыза алады, мысал ретінде файлдармен каталогтарды қорғаудың типтік жүйесін келтіруге болады. Бұл сияқты жүйе кез келген қолданушыға әрбір файлға және каталогқа қатынау құқығын беруді мүмкін етеді.

Ядроның басым режимге ауысуы арқылы қамтамасыз етілетін операциялық жүйенің бекемділігін арттыру, жүйелік шақыртулардың орындалу уақытын біраз бәсеңдету арқылы жүзеге асады. Басым ядроны жүйелік шақырту процессордың қолданушы режимінен басым режиміне ауысуын инициализациялайды, ал қосымшаға кері оралғанда – басым режимнен қолданушы режиміне ауысады. Процессорлардың барлық типтерінде ауысудың қосымша екі ретті ұстап қалуының әсерінен режимді ауыстыруы бар процедурасына өту режимді ауыстыруы жоқ процедураны шақыртудан баяуырақ орындалады.

Басым ядро және қолданушы режиміндегі қосымшалар негізінде тұрғызылған ОЖ архитектурасы шындығында да классикалық бола түсті. Оны көптеген танымал ОЖ қолданады: UNIX, VAX VMS, IBM OS/390, OS/2 және әртүрлі өзгешеліктері бар Windows NT.

Кейбір жағдайларда ОЖ құрастырушылар бұл классикалық архитектураның шегінен шығып, ядро мен қосымша жұмысын бір режимде ұйымдастырады. Novell компаниясының NetWare мамандандырылған ОЖ Intel x86/ Pentium процессорларының басым режимін ядро жұмысы үшін де, өздерінің ерекше қосымшалары NLM жүктелетін модульдерінің жұмысы үшін де қолданады. ОЖ бұл сияқты тұрғызғанда қосымшалардың ядроға қатынауы режимдерді ауыстыру сияқты емес, жылдам орындалады, бірақ дұрыс жұмыс жасамай тұрған қосымшалардан ОЖ модульдері алып тұрған жадыны қорғаудың берік аппараттық қорғаны жоқ. NetWare құрастырушылары өз операциялық жүйелерінің сенімділігін әлеуетті төмендетті, өйткені оның мамандандырылған қосымшаларының жиыны әрбір қосымшаны толық дұрыстаудың есебінен бұл архитектуралық кемшіліктің орнын жабады. Басым режим жұмысын мүлдем қолдамайтын процессорлар үшін әзірленген операциялық жүйелердің ядросымен қосымшалары да бір режимде жұмыс жасайды. Бұл сияқты процессорлардың мысалы ретінде IBM компаниясының процессорларына негіз болған Intel 8088/86 процессорын алуға болады.бұл компьютерлер үшін Microsoft компаниясы әзірлеген MS-DOS операциялық жүйесі жүйе ядросын құрайтын msdos.sys және io.sys модульдерінен тұрды (бірақ, бұл модульдер үшін «ядро» атауы қолданылмайтын, ал негізінен ядро болып табылатын). Оларға жүйелік шақыртуларымен command.com командалық интерпретатор, жүйелік утилиттер және қосымшалар қатынайтын. MS-DOS архитектурасы ОЖ архитектурасына сәйкес келеді. Дұрыс жазылмаған қосымшалар негізгі MS-DOS модульдерін бүлдіре алатын, тіптен, кейде ондай жағдайлар болып та тұратын, бірақ MS-DOS (және де сол сияқты MSX, CP/M сияқты ДК арналған ертеректегі операциялық жүйелер) қолдану аясы ОЖ сенімділігіне жоғары талаптар қоймайтын.

2.2.4 ОЖ тұрғызудың модульдік құрылымы.

ОЖ құрылымы модульді сипатта болады. ОЖ құрылымдаудың көп қолданатын тәсілі оның барлық модульдерін екі топқа бөлу:

– ядро – ОЖ негізгі қызметтерін атқаратын модульдер;

– ОЖ қосалқы функцияларын атқаратын модульдер.

ОЖ ядросының модульдері ОЖ келесі негізгі функцияларын орындайды: үдерістерді басқару, жадыны басқару, енгізу-шығару құрылғыларын басқару. ОЖ ядросы атқаратын функциялар жоғары жылдамдықты талап етеді және ол үшін үнемі жедел жадыда орналасады (резидентті модульдер).

Ядро құрамына есептеу үдерісін ұйымдастыратын ішкі жүйелік тапсырмаларды шешетін контексттерді ауыстыру, беттерді жүктеу/кері жүктеу, үзілімдерді өңдеу сияқты функциялар кіреді. Бұл функцияларға қосымшалар қатынай алмайды. Ядроның функцияларының келесі класы қосымшалар үшін қолданбалы бағдарламалық ортаны қалыптастырып, қосымшаларды сүйемелдейді. Қосымшалар ядроға қандай да бір әрекеттерді орындау үшін өздерінің сұратуларымен – жүйелік шақырулар- қатынай алады, мысалы, файлды ашу және оқу үшін, графикалық ақпаратты дисплейге шығару үшін, жүйелік уақытты алу үшін және т.б. Қосымшалармен шақыртылуы мүмкін ядро функциялары қолданбалы бағдарламалардың интерфейсі – API құрайды.

Ядро модулінің атқаратын функциялары операциялық жүйенің жиі қолданылатын функциялары болып табылады, сондықтан олардың орындалу жылдамдығы барлық жүйенің өнімділігін анықтайды. ОЖ жылдамдығын арттыруды қамтамасыз ету үшін ядроның барлық модульдері немесе олардың көп бөлігі жедел жадыда тұрады, яғни олар резидентті болып келеді.

Резидентті модуль операциялық жүйе жүктелгеннен кейін тұрақты түрде жедел жадыда сақталынады.

Қосалқы модульдер келесі функцияларды атқарады: ақпаратты архивтеу, дисктегі мәліметтерді дефрагментациялау, қажетті файлды іздеу және т.с.с.

ОЖ қосалқы модульдерін шартты түрде келесі топтарға бөлуге болады:

- Утилиттер – ОЖ басқару және сүйемелдеуінң жеке тапсырмаларын шешетін бағдарламалар;
- жүйелік өңдеуші бағдарламалар – мәтіндік және графикалық редакторлар, компиляторлар, құрастырушылар және т.с.с ;
- қолданушыға қосалқы қызмет түрлерін ұсынатын бағдарламалар – қолданушы интерфейсінің арнайы нұсқасы, калькулятор, ойындар және т.с.с.;
- процедуралар кітапханасы (жинағы) – қосымшаларды құрастыруды қысқартатын, әртүрлі тағайындаулары бар модульдер.

Утилиттер – қолданушыға сервистік қызметтерді ұсынатын қызмет етуші бағдарламалар. Негізінен олардың толық экранды, мәзір түрінде ұйымдастырылған қолданушымен өзара әрекеттесетін интерфейсі бар. Кейбір жағдайларда ғана интерфейс сұратулар түрінде ұйымдастырылады.

Мысалы, DOS құрамына әртүрлі мақсаттарға арналған ондаған утилиттер кіреді. Ал іс жүзінде олардың тек кейбіреулері ғана жиі қолданылады: FORMAT.COM, MODE.COM, KEYB.COM, FDISK.EXE, DISKCOPY.COM және т.б.

Windows утилиттерінің мысалдары: мәліметтерді архивтеу, жүйені қалпына келтіру, дискті дефрагментациялау, дискті тазарту, жүйе туралы мәлімет, Windows (Windows Update) жаңарту утилиті және т.б.

Қарапайым қосымшалар сияқты өз тапсырмаларын орындау үшін бағдарламаларды және ОЖ кітапханаларын өңдеуші утилиттер де жүйелік шақырулар арқылы ядро функцияларына қатынайды.

ОЖ ядроға және қосымша-модульдерге бөлу ОЖ оңай кеңейтілуін қамтамасыз етеді. Жоғары деңгейлі функцияны қосу үшін жаңа қосымшаны жасау жеткілікті және жүйе ядросын құрайтын жауапты функцияларды өзгертудің қажеті жоқ. Алайда, ядро функциясына өзгерістер енгізу қиынырақ болуы мүмкін, және бұл қиындық ядроның өзінің құрылымдық ұйымдастырылуына байланысты болады. Кейбір жағдайларда ядроны әрбір жөндеу оны толық қайта компиляциялауды талап етуі мүмкін.

Утилиттер, жүйелік өңдеуші бағдарламалар және түрінде рәсімделген ОЖ модульдері әдетте жедел жадыға тек өз функцияларын орындау уақытына ғана жүктеледі, яғни олар транзитті болып келеді. Жедел жадыда үнемі тек оның ядросын құрайтын ең қажетті ОЖ кодтары ғана болады. ОЖ бұл сияқты ұйымдастыру компьютердің жедел жадыны үнемдейді.

Ядро негізіндегі ОЖ архитектурасының маңызды қасиеті ядро функцияларын басым режимде орындаудың арқасында ОЖ кодтарын және мәліметтерін қорғау болып табылады.

2.3 Бақылау сұрақтары

2.3.1 ОЖ негізіне салынған негізгі тұжырымдамалар қандай?
2.3.2 Монолитті ядро және микроядроға қандай сипаттамалар тән?
2.3.3 Басым режимдегі ядро жұмысын сипаттаңыз.
2.3.4 ОЖ қандай модульдерден тұрады?

3 ҮДЕРІСТЕР МЕН АҒЫНДАР

3.1 Дәріс мақсаты

Дәріс мақсаты «үдеріс», «ағын», үдерістің күйі, ағындарды жоспарлау мен диспетчеризациялау, жоспарлау алгоритмі түсініктерімен таныстыру..

3.2 Теориялық мәліметтер

Есептеу машинасының қызметіне тікелей әсер ететін мультибағдарламалық ОЖ ішкі жүйелерінің бірі бұл үдерістер мен ағындарды басқарудың ішкі жүйесі. Ол үдерістер мен ағындарды құру, жою, олардың өзара әрекеттесуін қамтамасыз ету және жүйеде бір мезетте тұрған үдерістер мен ағындар арасында процессорлық уақытты бөлу қызметтерін атқарады.

Үдерістер мен ағындарды басқарудың ішкі жүйесі үдерістерді қажетті ресурстармен қамтамасыздандыру үшін жауап береді. ОЖ жадыда қай үдеріске қандай ресурс бөлінгендігін жазатын арнайы ақпараттық құрылымдарды бар. Ол үдеріске ресурсты дара пайдалануға немесе басқа үдерістермен бірлесе пайдалануға тағайындай алады. Кейбір ресурстар үдеріске ол құрылған кезде, ал кейбіреулері сұратуларды орындау барысында динамикалық түрде бөлінеді. Ресурстар үдерістің барлық өмір сүру уақытына немесе белгілі бір уақыт аралығына үдеріске тіркеліп қойылуы мүмкін. Үдерістерді басқарудың ішкі жүйесі бұл функцияларды орындау барысында жадыны басқарудың ішкі жүйесі, енгізу-шығарудың ішкі жүйесі, файлдық жүйе сияқты ресурстарды басқаруға жауапты ОЖ басқа да үшкі жүйелерімен өзара әрекеттеседі.

Үдерістер мен ағындарды басқарудың ішкі жүйесінің маңызды функцияларының бірі ағындарды синхронизациялау болып табылады.

3.2.1 «Үдеріс», «ағын» түсініктері

Мультибағдарламалауды қамту үшін ОЖ араларында процессор және басқа да компьютер ресурстары бөлінісілетін ішкі жұмыс бірліктерін анықтап, рәсімдеуі керек. Қазіргі уақытта ОЖ көбінде жұмыс бірлігінің екі типі анықталған.

Үдеріс ретінде орындалу кезеңінде тұрған бағдарламаны түсінуге болады. Сонымен қатар, үдерісті процессор үшін жұмыс бірлігі ретінде де қарастыруға болады. Жаңа процессорлар үшін одан да кіші жұмыс бірлігі бар, ол ағын немесе тармақ. Басқаша айтсақ, бір үдеріс бір немесе бірнеше ағын туындата алады.

Үдерістер де, ағындар да бар ОЖ, ОЖ үдерісті процессорлық уақыттан басқа барлық ресурс түрлерін пайдалануға сұраным ретінде қарастырылады. Аталған маңызды ресурс процессорлық уақыт басқа жұмыс бірліктері – ағындардың арасында үлестіріледі. Ағын деген атау олар командалар тізбегі (командалар бірінен соң бірі) болғандықтан берілген.

Ең қарапайым жағдайда үдеріс бір ағыннан тұрады, 80-жылдардың ортасына дейін (мысалы, UNIX ертеректегі нұсқаларында) «үдеріс» түсінігі осылай түсіндірілген, және кейбір заманауи ОЖ осы түрінде сақталынған.

Үдерістер ресурстарды үлестіруге араласа алмауы және бір-бірінің кодтарымен мәліметтерін бүлдіре алмауы үшін ОЖ маңызды тапсырмасы үдерістерді бір-бірінен оқшаулау болып табылады. Ол үшін ОЖ әрбір үдерісті жеке виртуалды адрестік кеңістікпен қамтамасыз етеді, сондықтан ешбір үдеріс басқа үдерістің командалары мен мәліметтеріне тікелей қатынай алмайды.

Үдерістің виртуалды адрестік кеңістігі – бұл үдерістің бағдарламалық модулі манипуляция жасай алатын адрестер жиыны.

ОЖ үдерістің виртуалды адрестік кеңістігін үдеріске бөлінген физикалық адресте бейнелейді. Өзара әрекеттесу барысында үдерістер ОЖ қатынайды, ал ол денекерші қызметін атқара отырып, үдерістерге үдеріс аралық байланыс құралдарын – конвейерлер, пошта жәшіктері, жадының бөлінісілетін секцияларын және т.б. ұсынады.

Ағындар операциялық жүйелерде есептеулерді параллельдеу құралдары ретінде пайда болды. Әрине, бір бағдарлама аясында есептеудерді параллельдеу міндеттерін дәстүрлі тәсілдермен де шешуге болады.

Біріншіден, қолданбалы бағдарламалаушы параллелизмді ұйымдастырудың күрделі міндетін қосымша ішінде басқаруды есептеулердің тармақтарына кезек-кезек (периодты түрде) беріп отыратын диспетчер ішкі бағдарламасын бөліп алу арқылы шеше алады. Бұл жағдайда бағдарлама басқаруды берулері көп логикалық жағынан өте шатасқан болып шығады, ал бұл өз кезегінде бағдарламаның ретке келтіліруін және өзгертілуін қиындатады.

Екіншіден, мұның тағы бір шешімі бір қосымшаға әрбір параллельді жұмыс үшін бірнеше үдеріс құру. Бірақ, үдерісті құруда стандартты құралдарды пайдалану бұл үдерістердің бір тапсырманы шешетінін, яғни олардың арасында ортақ нәрселер көп екендігін есепке алуға мүмкіндік бермейді. Олар бір мәліметтермен жұмыс жасап, бір сегментті кодты пайдаланып және есептеу жүйесінің ресурстарына да қатынау құқықтары бірдей берілуі мүмкін. Одан басқа, ОЖ әрбір үдерісті құруға қандайда бір ресурстарды жұмсайды, ал бұл жағдайда олар жөнсіз қайталанылады - әрбір үдеріске өзінің виртуалды адрестік кеңістігі, физикалық жадысы беріліп, енгізу-шығару құрылғылары бекітіледі және т.с.с.

Жоғары аталғандардан ОЖ үдерістен басқа бір қосымшаның есептеулерінің жеке тармақтары арасындағы тығыз байланысты есепке алатын басқа есептеулерді параллельдеу механизмі қажет екендігі шығады. Бұл мақсатқа жету үшін заманауи ОЖ көп ағынды өңдеу (multithreading) механизмін ұсынады. Бұл жағдайда жаңа жұмыс бірлігі енеді – орындалу ағыны, ал «үдеріс» түсінігі өз мағынасын өзгертеді. «Ағын» түсінігіне процессордың бағдарламаның бір командасынан екінші командасына тізбектеле ауысуына сәйкес келеді. ОЖ ағындар арасында процессорлық уақытты бөледі. ОЖ үдеріске оның ағындары ортақ пайдаланылатын адрестік кеңістікті және ресурстар жиынын тағайындайды.

Мультибағдарламалауды үдеріс деңгейіне қарағанда ағындар деңгейінде қолданған нәтижелі болады. Әрбір ағынның өз командалар есептегіші және стекы бар. Бір үдеріс аясында бірнеше ағын түрінде рәсімделген есептің жекелеген бөліктерін жалған параллельді (псевдопараллельді, немесе мультипроцессорлы жүйедегі параллельді) орындау арқылы жылдам орындауға болады. Мысалы, егер электронды кесте көп ағынды өңдеудің мүмкіндіктерін ескере отырып құрастырылған болса, онда қолданушы өз жұмыс бетін қайта есептеуді сұрата отырып, сол мезетте кестені толтыра алады. Көп ағындылықты әсіресе үлестірілген қосымшаларда қолданған аса тиімді болады, мысалы, көп ағынды сервер бір мезетте бірнеше қолданушылардың сұратуларын орындай алады.

Ағындарды қолдану тек параллельді есептеудің арқасында жүйе өнімділігін арттырумен ғана емес, сонымен бірге логикалық, читабельдік жағынан жоғарырақ бағдарламалар құру мақсатында да жүреді. Бірнеше орындалу ағынын қатар алып отыру бағдарламалауды қарапайымдатады. Мысалы, «жазушы-оқушы» типтегі тапсырмаларда бір ағын буферге жазуды, ал келесісі буфердан оқуды орындайды. Екеуі бір буферді бөлісіп отырғандықтан оларды жеке үдерістер етіп жасаудың қажеті жоқ. Ағындарды қолданудың тағы бір мысалы – пернетақтадан енгізілген үзулер (del немесе break) сияқты сигналдары басқару. Бір ағын үзу сигналдарын өңдеудің орнына емес, сигналдың түсуін тұрақты тосып тұруға тағайындалады. Осылай, ағындарды пайдалану қолданушы деңгейіндегі үзілімдер қажеттілігін азайта алуы мүмкін. Бұл мысалдарда параллельді орындаудың өзі емес, u1087 бағдарламасының айқындылығы маңызды.

Көп ағынды өңдеуді ағындары, тіпті бір үдеріске жататын ағындары да түрлі процессорларда шындығында да параллельді (жалған параллельді емес) орындалатын мультипроцессорлық жүйелерде қолданған нәтижелі болады.

3.2.2 Үдерістер мен ағындарды құру

Операциялық жүйе есепте тұрған үдерістерге жаңа үдерісті қосу үшін ол мәліметтер құрылымын жасайды. Мәліметтер құрылымы үдерісті басқаруда қолданылады және оны негізгі жадының адрестік кеңістігіне орналастырады. Осы әрекеттер негізінде жаңа үдеріс құрылады.

Пакеттік өңдеу ортасында үдеріс тапсырма келіп түскенде жауап ретінде құрылады; интерактивті ортада үдеріс жаңа қолданушы ортаға кіруге талпынған уақытта пайда болады. Екі жағдайда да жаңа үдерісті құруға жауаптылық операциялық жүйенің мойнында болады. Мұнан басқа, операциялық жүйе қосымшаның талабы бойынша да үдеріс құра алады. Мысалы, егер қолданушы файлды басып шығаруға сұрату жіберсе, онда операциялық жүйе басып шығаруды басқаратын үдеріс құра алады. Одан кейін, басып шығаруды сұратқан сұрату, баспаға кететін уақытқа қарамастан өз жұмысын жалғастыра алады.

Дәстүрлі түрде операциялық жүйе үдерісті қолданушылармен қосымшалардың елеуінсіз құрады; бұл сияқты тәсіл көптеген заманауи операциялық жүйелерде қалыптасқан. Бірақ кейде бір үдеріс келесі үдерістің құрылуына себепші болуы керек болады. Мысалы, қосымша үдерісі жаңа үдерісті генерациялайды, ал бұл үдеріс бірінші үдерістен мәліметтерді алып, оларды әрі қарай талдауға керекті ыңғайлы түрге келтіреді. Жаңа үдеріс қосымшамен параллельді жұмыс жасайды және ара арасында жаңа мәліметтерді алу үшін белсенді болады. Бұл сияқты ұйымдастыру қосышаны құрылымдауда аса пайдалы болады. Егер операциялық жүйе жаңа үдерісті басқа үдерістің сұратуы бойынша жасаса, бұл үдерісті туындату (process spawning) деп аталады.

Бір үдеріс жаңа үдерісті туындатса, жаңа үдерісті туындатқан үдеріс - аталық (parent), туындалған үдеріс – еншілі немесе тұқым (child) деп аталады. Жалпы «туысқан» үдерістер өзара ақпарат алмасып, бір-бірімен әрекеттеседі.

Үдерісті құру - бұл ең бірінші, үдеріс сипаттауышын (handle – дескриптор) құруды білдіреді. Үдеріс сипаттауышы ретінде ОЖ үдерісті басқаруға қажетті үдеріс туралы барлық мәліметтер бар бір немесе бірнеше ақпараттық құрылым пайдаланылады. Бұл сияқты мәліметтер қатарына мысалы, үдеріс идентификаторы, орындалатын модульдің жадыда орналасуы туралы мәлімет, үдерістің басымдылық деңгейі (басымдылық және құтынау құқықтары) және т.с.с. жатады. Үдеріс сипаттауышының мысалдары: OS/360-дегі тапсырмаларды басқару блогы (TCB – Task Control Block), OS/2-де процессордың басқарушы блогы (PCB – Process Control Block), UNIX-те үдеріс дескрипторы, Windows NT-де объект-үдеріс (object-process).

Үдерісті құруға осы үдерістің орындалатын бағдарламасының коды мен мәліметтерін дисктен жедел жадыға жүктеу кіреді. Бұл үшін ОЖ бағдарламаның дисктегі орнын тауып, жедел жадыны қайта үлестіріп және жаңа үдерістің орындалатын бағдарламасына жады бөліп беруі керек. Одан кейін бағдарламаны оған бөлінген жады аумақтарына қайта оқуы керек және мүмкін болса, бағдарламаны жадыда орналасқанына байланысты баптау керек болады.

Көп ағынды жүйеде ОЖ үдерісті құру барысында әрбір үдеріс үшін минимум бір орындау ағынын тұрғызады. Ағынды құрғанда да ОЖ үдерісті құрғандағы сияқты арнайы ақпараттық құрылым – ағын сипаттауышын генерациялайды. Ағын сипаттауышында ағын идентификаторы, қатынау құқықтары мен басымдылықтар туралы мәлімет, ағынның күйі және т.б. ақпараттар болады. Бастапқы күйде ағын (немесе үдеріс, егер «ағын» түсінігі анықталмайтын жүйе туралы айтылса) тоқтатылған күйде болады. Ағынды орындауға таңдап

алу сол сәтте жүйеде тұрған барлық ағындар мен үдерістерді есепке ала отырып, жүйеде қалыптасқан процессорлық уақытты беру ережелеріне сай жүргізіледі.

Үдерістерді басқару барысында операциялық жүйе ақпараттық құрылымның негізгі екі типін пайдаланады: үдеріс дескрипторы және үдеріс контекстісі. Үдеріс дескрипторында ядроға үдерістің барлық өмір циклында, яғни оның активті әлде пассивті күйде ме, үдеріс бейнесі жедел жадыға жүктелген бе алде дискке кері жүктелген бе соған қарамастан қажет болатын ақпарат бар.

Жекелеген үдерістер дескрипторлары үдерістер кестесін құрайтын тізімге біріктірілген. Үдерістер кестесіне жады динамикалық түрде ядро аймағынан беріледі. Үдерістер кестесіндегі тұрған ақпараттың негізінде операциялық жүйе үдерістерді жоспарлау мен оларды синхронизациялауды жүргізеді. Дескрипторда үдерістің күйі, жедел жадыда және дискіде үдеріс бейнесінің орналасуы, басымдылықтардың кейбір құраушыларының мәндері туралы, сонымен қатар оның ауқымды басымдылықтағы алатын орны, үдерісті құрған қолданушының идентификаторы, тектес үдерістер туралы, бұл үдеріс орындалуын күтіп тұрған оқиғалар туралы және т.б. ақпарат болады.

Үдеріс контекстісі үдерістің орындалуы үзілген жерден оның орындалуын қалпына келтіруге қажетті үдеріс туралы ақпараттың жеделділігі төменірек, бірақ көлемі жағынан үлкенірек бөлімін сақтайды: процессор регистрлерінің ішіндегісі, процессор орындайтын жүйелік шақырулардың қателіктер коды, бұл үдеріс ашқан барлық файлар мен аяқталмаған енгізу-шығару операциялары туралы ақпарат және үзіліс басталған мезеттегі есептеу ортасының күйін сипаттайтын басқа да мәліметтер. Контекст те үдеріс дескрипторы сияқты ядро бағдарламаларына ғана қол жетімді, яғни ол ОЖ виртуалды адрестік кеңістігінде орналасады, бірақ ол ядро аймағында сақталмайды, үдеріс бейнесіне жалғасып жатады және онымен бірге жылжытылады, егер қажет болса жедел жадыдан дискіге көшіріледі.

3.2.3 Үдерістерді аяқтау

Кез келген компьютерлік жүйеде үдерістің орындалуы аяқталды ма, жоқ па соны анықтауға мүмкіндік беретін құралдар болуы керек. Пакеттік тапсырмада halt («аялдау») типті команда болуы керек, немесе үдерісті аяқтауға әкелетін ОЖ қызметін анық шақырту болуы керек. Бірінші жағдайда ОЖ үдерістің аяқталғандығы туралы хабардар ету үшін үзілу генерацияланады. Мысалы, уақытты бөлісуі бар жүйеде қолданушы үдерісі қолданушы жүйеден шыққанда немесе терминалды өшіргенде аяқталуы керек. Дербес компьютерде немесе жұмыс станциясында қолданушы қосымшадан шыға алады (мысалы, мәтінді немесе электронды кестені өңдеу бағдарламасын өшіру). Бұл әрекеттердің барлығы да ақыр соңында ОЖ үдерісті аяқтайын қызметінің шақырылуына әкеледі.

Қарапайым аяқтау – үдеріс ОЖ қызметін ол өзінің жұмысын аяқтап болғандығы туралы айту үшін шақырады.

Жады көлемінің жеткіліксіздігі – үдерістің жұмысы үшін жүйеде бар жады көлемінен үлкенірек жады қажет.

Берілген жады аймағынан шығып кету – үдеріс өзінде қатынау құқығы жоқ жады ұяшығына қол жеткізуге тырысады.

Қорғаныс қателігі – үдеріс өзіне қол жетімсіз ресурсты немесе файлды қолдануға талпынады немесе мұны рұқсат етілмеген жолмен істеуге тырысады, мысалы, тек оқу үшін ғана ашылған файлға жазбалар енгізуге тысыру.

Арифметикалық қате – үдеріс рұқсат етілмеген арифметикалық операцияны істеуге талпынды, мысалы, нөлге бөлу, немесе аппараттық қамтамасыздандырудың мүмкіншіліктерінен жоғары сандарды қолдануға талпыну.

Енгізу-шығару қателіктері – енгігу немесе шығару барысында қателіктер кетеді, мысалы, қажетті табылмайды немесе мүмкін емес операцияны іске асыруға талпыныс жасалады (мысалы, баспадан шығару құрылғысынан оқу).

Дұрыс емес команда – үдеріс жоқ команданы орындауға тырысады.

Қол жетімсіз басымдылықтары бар командалар – үдеріс ОЖ үшін сақталынған (зарезервированный) командаңы қолдануға тырысады.

Оператор немесе ОЖ араласуы – қандай да бір себептермен ОЖ үдерісті аяқтауы мүмкін (мысалы, өзара бұғатталған жағдайында).

Аталық үдерістің аяқталуы – аталық үдерістің аяқталған уақытында ОЖ оның барлық туынды (дочерние) үдерістерін де автоматты түрде тоқтатуы мүмкін.

Аталық үдерістің сұрату салуы – жалпы аталық үдерістің өзінің кез келген туынды үдерісін тоқтатуға құқысы бар.

3.2.4 Ағынның күйі

ОЖ ағындарды жобалағанда олардың күйін есепке алады. Мультибағдарламалық жүйеде ағын негізгі үш күйдің бірінде тұра алады:

- орындалу – ағынның бенсенді күйі, бұл мезетте ағында барлық қажетті ресурстар болады және үдеріс нақты өзін орындап жатады;
- күту – ағынның пассивті күйі, бұл мезетте ағын өзінің ішкі себептеріне байланысты оқшауланады (енгізу-шығару операцияларының аяқталуын күтуде, басқа ағыннан хабарлама алу және т.б.);
- дайын - ағынның пассивті күйі, бірақ бұл күйінде ағын оған қатысты сыртқы жағдайларға байланысты оқшауланады (барлық қажетті ресурстар бар, орындалуға дайын бірақ процессор басқа ағынды орындап жатыр).

Ағын өзінің барлық ғұмырында ОЖ қабылданған ағындарды жоспарлау алгоритміне сәйкес бір күйден келесі күйге өтеді.

Ағынның әдіттегі күй графын қарастырайық. Жаңа ғана құрылған ағын дайындық күйінде тұр, ол орындалуға дайын және процессорға кезекте тұр. Жоспарлаудың нәтижесінде ағындарды басқарудың ішжүйесі ағынды белсендендіруге (қосуға) шешім қабылдаған мезетте, ол орындалу күйіне өтеді және сол күйде немесе ол өзі процессорды өзі босатып, басқа бір оқиғаны күтуге өткенге дейін немесе процессордан өзін ықтиярыз ығыстырып шығарғанға дейін болады, мысалы оған бөлінген процессорлық уақыт квантінің аяқталуының нәтижесінде. Соңғы жағдайда ағын дайындық күйіне қайтіп оралады. Ағын күту күйінен кейін де, күткен оқиғасы орындалғаннан соң, бұл күйіне қайтадан оралады.

Бірпроцессорлы жүйеде орындалу күйінде тек бір ғана ағын бола алады, күту және дайындық күйлерінің әрқайсысында бірнеше ағын бола алады. Бұл ағындар сәйкесінше күтудегі және дайын үдерістердің кезегін құрайды. Ағындар кезегі жекелеген ағындардың сипаттамаларын тізімге біріктіру арқылы ұйымдастырылады. Сонымен, әрбір ағынның сипаттамасында негізгі ақпараттан басқа тым болмағанда өзімен кезекте қатар тұрған ағынның сипаттамасына бір көрсеткіш болады. Кезекті бұл сияқты ұйымдастыру оларды оңай қайта ретке келтіруге, ағындарды қосуға, алып тастауға, ағындарды бір күйден екніші күйге оңай ауыстыруға мүмкіндік береді.

3.2.5 Үдерістерді жоспарлау

Үдерістің барлық өмір сүрген уақыты ішінде оның ағындарының орындалуы бірнеше рет үзіліп, бірнеше рет жалғасуы мүмкін. Бір ағынды орындаудан екінші ағынға өткен жоспарлаумен дипетчеризацияның арқасында жүргізіледі. Ағымдағы белсенді ағынның жұмысын қай сәтте тоқтату керек және орындалу мүмкіндігін қай ағынға беру керектігін анықтау жұмыстары жоспарлау деп аталады. Ағындарды жоспарлау үдерістермен ағындардың сипаттамаларында жатқан ақпараттардың негізінде жүргізіледі. Жобалау барысында ағындардың басымдылығы, олардың кезекте күту уақыты, жинақталған орындалу уақыты, енгізу-шығаруға жүгіну қарқындылығы (интенсивность обращений к вводу-выводу) және тағы басқа факторлар есепке алынады. ОЖ ағындар бір үдеріске ме әлде бірнеше үдеріске тиесілі ме оған қарамастан ағындардың орындалуын жоспарлайды. Мысалы үшін,

ОЖ қандай да бір үдерістің ағынын орындап болғаннан соң, орындауға сол үдерістің басқа бір ағынын таңдап ала алады немесе басқа үдерістің ағынын орындауға тағайындауы мүмкін.

Келтірілген тапсырмаларды шешетін ағындарды жоспарлаудың түрлі алгоритмдері бар. Жоспарлау алгоритмдері түрлі мақсаттарды көздейді және мультибағдарламалаудың түрлі сапасын қамтамасыз етеді.

Көптеген әмбебап (түрлі қажеттіліктерге арналған) ОЖ жоспарлау динамикалық түрде жүргізіледі, яғни жүйе жұмыс жасап тұрған мезетте ағымдағы жағдайды талдай келе шешім қабылданады. ОЖ белгісіздік (анықталмағандық) жағдайында жұмыс жасайды – ағындармен үдерістер кездейсоқ мезетте пайда болады және солай болжалмай (алдын ала белгісіз мезетте) аяқталады. Бұл сияқты белгісіздік шартында (жағдайында) тапсырмалардың орындалуының қандай да бір тиімді ретін табу үшін ОЖ бірталай (бірқатар, едәуір) ресурстар қажет болады.

Жоспарлаудың тағы бір түрі – статикалық – бұл түрі бір мезетте орындалатын тапсырмалардың барлық жиыны алдын ала анықталған мамандандырылған (арнайы) жүйелерде қолданылады, мысалы, нақты уақыт жүйелерінде. Жоспарлаушы статикалық деп аталады, егер ол жоспарлау бойынша шешімдерді жүйе жұмыс жасап тұрғанда емес, алдын ала қабылдап қойған болса. Статикалық жоспарлаушының жұмыс нәтижесі процессор қай ағынға/үдеріске қашан және қанша уақытқа берілуі керектігі көрсетілген кесте түрінде болады, және оны жұмыс кестесі деп атайды.

Диспетчерлендіру дегеніміз жоспарлаудың (динамикалық немесе статистикалық) нәтижесінде табылған шешімді жүзеге асыру, яғни процессорды бір ағыннан келесі ағынға ауыстыру. Ағын жұмысының орындалуын тоқтату үшін ОЖ оның контекстін есте сақтап алады, ол бұл ақпаратты осы ағынның орындалуын қалпына келтіру үшін пайдаланады.

Диспетчерлендіру мыналарға келіп тіреледі:

- ауыстыру қажетті ағымдағы ағынның контекстін сақтауға;
- жаңа ағынның контекстін жүктеуге;
- жаңа ағынды орындауға қосу.

Контекстерді ауыстыру операциясы есептеу техникасының өнімділігіне анық әсер етеді, сондықтанда ОЖ ағындарды диспетчерлендіруді процессордың аппараттық құралдармен бірлесе отырып жүргізеді. Ағын контекстісінің бұл үдерістің барлық ағындарына ортақ бөлігі (ашық файлдарға сілтемелер) және тек осы ағынның өзіне ғана тиесілі бөлігі (регистрдің мазмұны, командалар есептеуіші, процессор режимі) бар.

Үдерістерді жоспарлауға келесі тапсырмалар кіреді:

- орындалатын үдерісті ауыстыру мезетін анықтау;
- дайын үдерістердің кезегінен орындауға үдерісті таңдау;
- «ескі» және «жаңа» үдерістің контекстісін ауыстыру.

Алғашқы екі тапсырма бағдарламалық құралдармен, ал соңғысы көбірек аппараттық құралдармен шешіледі.

Жоғарыда аталған тапсырмаларды түрлі жолдармен шешетін, әртүрлі мақсаттарды көздейтін және мультибағдарламалаудың түрлі сапасын қамтамасыз ететін үдерістерді жоспарлаудың түрлі алгоритмдері бар. Бұл көптеген алгоритмдердің арасынан жиі кездесетін алгоритмдердің екі тобын қарастырайық: кванттаудың негізіндегі алгоритмдер және басымдылықтар негізіндегі алгоритмдер.

Кванттау негізіндегі алгоритмдерге сәйкес, белсенді үдеріс ауыстыру жүргізіледі, егер,

- үдеріс аяқталса және жүйеден кетсе;
- қателік орын алса;
- үдеріс КҮТУ күйіне өтсе;
- бұл үдеріске бөлінген процессорлық уақыт кванты таусылса.

Өз квантын тауысқан үдеріс ДАЙЫНДЫҚ күйіне ауысады және өзіне жаңа процессорлық уақыт кванты берілгенін тосады, ал орындалуға дайын үдерістер кезегінен

қандай да бір таңдау ережесіне сәйкес жаңа үдеріс таңдап алынады. Осылай ешбір үдеріс процессорды ұзақ ұстап тұрмайды, сондықтанда уақытты бөлу жүйелерінде кванттау кең қолданыс алуда.

Үдерістерге бөлінген кванттар барлық үдерістер үшін бірдей немесе әртүрлі болады. Бір үдеріске бөлінген квант тиянақталған көлемде болуы мүмкін немесе үдерістің өмір сүруінің әр периодында өзгерулері мүмкін. өздеріне бөлінген квантты толық пайдаланбаған (мысалы, енгізу-шығару операцияларына кетуіне байланысты) үдерістер келесі қызмет көрсетуде басымдылық түрінде компенсация (өтеу, өтемақы) алуы да, алмауы да мүмкін.

Алгоритмдердің келесі тобы үдерістің «басымдылығы» түсінігін пайдаланады.

Басымдылық – үдерістің есептеу машинасының ресурстарын қолданудағы басымдылық (артықшылық) деңгейін сипаттайтын сан, жекелей алғанда: процессорлық уақытты, басымдылық жоғары болған сайын артықшылық (басымшылдық) жоғары болады.

Басымдылық бүтін немесе бөлшек, оң немесе теріс мән арқылы көрсетіледі. Үдерістің басымшылығы жоғары болған сайын, ол кезекте соғұрлым аз уақыт өткізеді. Басымдылық жұмыстың маңыздылығына немесе енгізілген ақыға байланысты директивті түрде жүйе әкімдерімен тағайындалады, я болмаса қандайда бір ережелерге сәйкес ОЖ өзі есептеуі мүмкін үдерістің бүкіл өмірі бойында тиянақталған (тұрақты) болып қалуы немесе қандайда бір заңдылыққа сәйкес уақыт бойында өзгеріп тұруы мүмкін. Соңғы жағдайда басымдылық динамикалық деп аталады.

Басым алгоритмдердің екі түрі бар: салыстырмалы басымдылықтарды пайдаланатын алгоритмдер және абсолюттік басымдылықтарды пайдаланатын алгоритмдер.

Екі жағдайда да кезектегі дайын үдерістердің ішінен орындауға үдерісті таңдау бірдей жүргізіледі: ең жоғарғы басымдылығы бар үдеріс таңдап алынады.

Белсенді үдерісті алмастыру мезетін анықтау мәселесі де әртүрлі шешіледі. Салыстырмалы басымдылықты жүйелерде белсенді үдеріс өзі КҮТУ күйіне өтіп, процессорды өзі босатқанға дейін орындалады (немесе қандайда бір қателік орын алады, немесе үдеріс аяқталады). Абсолюттік басымдылықты жүйелерде белсенді үдерістің орындалуы тағы бір шарт орындалғанда үзіледі: егер дайын үдерістер кезегінде басымдылығы белсенді үдерістің басымдылығынан жоғарырақ үдеріс пайда болған жағдайда. Бұл жағдайда үзілген үдеріс дайындық күйіне ауысады. Көптеген ОЖ жоспарлау алгоритмдері кванттауды да, басымдылықтарды да қолданылып тұрғызылған. Мысалы, жоспарлаудың негізінде кванттау жатады, бірақ квант көлемі және дайындардың арасынан үдерісті таңдау үдерістердің басымдылықтары бойынша жүргізіледі.

Барлық көптеген жоспарлау алгоритмдерін бір–бірінен түбегейлі әртүрлі (бөлек) 2 класқа бөлуге болады: ығыстыратын және ығыстырмайтын.

- ығыстырмайтын (non-preemptive) алгоритмдерде белсенді ағын өз қалауымен басқаруды ОЖ ол кезектен басқа орындалуға дайын ағынды таңдап алуға бергенге дейін белсенсі ағынға орындалып тұра беру мүмкіншілігі берілген.

- Ығыстыратын (preemptive) алгоритмдер –ағындарды жоспарлаудың бұл тәсілінде процессорды бір ағынды орындаудан келесі ағынды орындауға ауыстыру белсенді тапсырмамен емес, ОЖ қабылданады.

Алгоритмдердің екі типінің арасындағы басты айырмашылық ағындарды жоспарлау механизмінің орталықтандырылу деңгейінде. Ығыстыратын алгоритмдер толығымен ОЖ шоғырланған және бағдарламалаушы өз бағдарламасының басқа тапсырмалармен қатар орындалатынына алаңсыз өз бағдарламасын жаза береді. Бұл жерде ОЖ белсенді ағынды орындаудан алып тастау уақытын өзі анықтайды, оның контекстін есте сақтап, кезектегі дайындардың келесісін таңдап, оның контекстісін жүктей отыра жаңа ағынды орындауға іске қосады.

Ығыстармайтын мультибағдарламалауда жоспарлау механизмі ОЖ мен қолданбалы бағдарламалар арасында үлестірілген. Қолданбалы бағдарлама ОЖ басқаруды

алғаннан кейін қажет деп тапқан уақытында қандайда бір жүйелік шақырудың көмегімен басқаруды қайтадан ОЖ береді. ОЖ ағындардың кезегін құрайды және орындауға келесі ағынды қандайда бір ережеге (мысалы, басымдылықты ескере отырып) сәйкес таңдап алады.

Бағдарламаларды жоғарғы өнімді орындауға бағдарланған барлық дерлік заманауи ОЖ (UNIX, Windows NT/2000, OS/2, VAX/VMS) ағындарды (үдерістерді) жоспарлаудың ығыстыратын алгоритмі жүзеге асырылған.

3.3 Бақылау сұрақтары

3.3.1 Үдеріс пен ағынның анықтамасын беріңіз.

3.3.2 Үдерістер мен ағындарды құру және аяқтау қалай жүргізіледі?

3.3.3 Үдеріс қандай күйлерде бола алады?

3.3.4 Үдерістерді жоспарлау қалай жүргізіледі?

3.3.5 Кванттау негізіндегі алгоритмдер басымдылық негізіндегі алгоритмдерден несімен ерекшеленеді?

3.3.6 Жоспарлаудың ығыстыратын алгоритмдері ығыстырмайтын алгоритмдерден айырмашылығы қандай?

4 ЖАДЫНЫ БАСҚАРУ

4.1 Дәріс мақсаты

Дәріс мақсаты ОЖ жадыны басқару функциялары және жадыны ұйымдастыру тәсілдерімен танысу.

4.2 Теориялық мәліметтер

Бұл жерде жады (memory) ретінде компьютердің жедел жадысы аталып тұр. Сыртқы жады (storage) деп атайтын қатты диск жадысымен салыстырғанда жедел жадыда ақпаратты сақтау үшін үнемі тоққөзі қажет.

Жады мультибағдарламалық ОЖ тарапынан мұқият басқаруды талап ететін маңызды ресурс болып табылады. Жадының ролінің ерекшелігін келесі фактімен түсіндіруге болады: процессор бағдарламалардың нұсқаулықтарын тек олар жадыда тұрғында ғана орындай алады. Жады қолданбалы бағдарламалар арасында да, ОЖ өз модульдерінің арасында да үлестіріледі.

4.2.1 ОЖ жадыны басқару бойынша функциялары

Мультибағдарламалаудың пайда болуымен ОЖ алдында бір мезетте орындалып жатқан бірнеше бағдарлама арасында бар жадыны үлестірумен байланысты жаңа тапсырмалар пайды болды.

Мультибағдарламалық жүйеде жадыны басқару бойынша ОЖ функциялары:

- бос және бос емес жадыны бақылап отыру;
- үдерістерге жадыны бөлу және үдерістер аяқталғаннан соң жадыны босату;
- негізгі жадының көлемі онда барлық үдерістерді орналастыруға жеткіліксіз болғанда үдерістердің кодтарымен мәліметтерін жедел жадыдан дискке ығыстыру (толық немесе жартылай) және жедел жадыда орын босағанда оларды кері әкелу;
- бағдарламаның адресін физикалық жадының нақты аусағына баптау;
- жадыны дефрагментациялау;
- жадыны қорғау, мұның мәні төмендегідей: орындалып жатқан үдеріске басқа үдеріске бөлінген жадыға мәлісеттерді жазғызып, оқытпау. Көбінесе бұл функция ОЖ бағдарламалық модульдері аппараттық құралдармен өзара тығыз байланысқа түскенде жүзеге асырылады.

4.2.2 Адрестердің типтері

Бағдарламаның өмірлік циклінің түрлі деңгейлерінде айнымалылармен командаларды идентификациялау үшін символдық (белгілік) аттар (белгі), виртуалды адрестер және физикалық адрестер қолданылады (сурет 4.1).

Сурет 4.1 – Адрестердің типтері

- Символдық аттарды қолданушы алгоритмдік тілде немесе ассемблерде бағдарламалар жазған уақытта меншіктейді (береді).

- Виртуалды адрестер, кей уақыттарда математикалық немесе логикалық адрестер деп те аталады. Оларды бағдарламаны машиналық тілге аударатын транслятор шығарады. Жалпы алғанда трансляциялау барысында бағдарлама жедел жадының қай жеріне жүктелетіні белгісіз, онда транслятор бағдарламаның бастапқы адресі нолдік адрес болады деп, афнймалылармен командаларға виртуалды (шартты) адрестер береді.

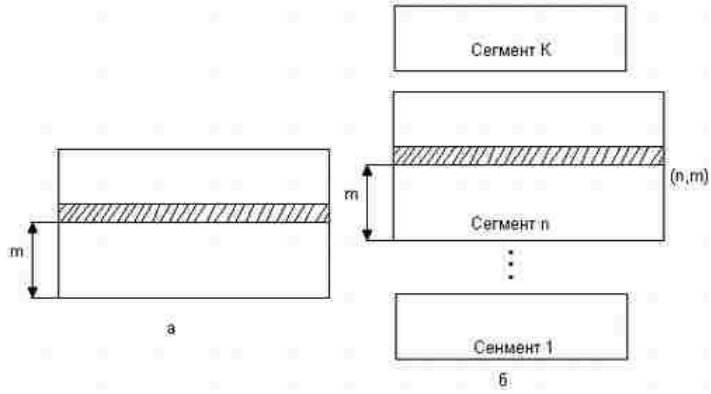
- Физикалық адрестерге айнымалыламен командалар шындығындада орналасқан жедел жады ұяшықтарының номерлері сәйкес келеді.

Үдерістің виртуалды адрестер жиынтығы виртуалды адрестік кеңістік деп аталады. Барлық үдерістердің виртуалды адрестік кеңістігінің мүмкін болатын диапазоны бір болады. Мысалы, 32-разрядтық виртуалды адрестерді қолданғанда бұл диапазон 00000000_{16} және $FFFFFFFF_{16}$ шектерімен беріледі. Олай болса да, әрбір үдерістің өз виртуалды адрестік кеңістігі болады – транслятор айнымалылармен әрбір бағдарламаның кодына тәуелсіз виртуалды адрестер тағайындайды. Айнымалылар мен командалардың адрестерінің бірдей болып кетуі шиеленіске әкелмейді, бұл сияқты жағдайда бұл айнымалылар бір мезетте жадыда болады, ОЖ оларды әртүрлі физикалық адрестерде бейнелейді.

Әртүрлі ОЖ виртуалды адрестік кеңістікті құрылымдаудың түрлі тәсілдері қолданылады. Кейбір ОЖ үдерістің виртуалды адрестік кеңістігі физикалық жадыға ұқсайды, виртуалды адрестердің үзіліссіз тізбегі ретінде бейнеленген. Адрестік кеңістіктің бұл сияқты құрылымын жазық (flat) деп те атайды.

Виртуалды адрестік кеңістіктің басынан (көбінесе бұл мән 000...000) жылжуды көрсететін жалғыз сан – виртуалды адрес болып табылады (сурет 4.2 а). Адрестің бұл сияқты түрі сызықтық виртуалды адрес деп аталады.

Басқа ОЖ виртуалды адрестік кеңістік (немесе секция, немесе облыс, немесе тағы басқа терминдермен аталады) сегмент деп аталатын бөліктерге бөлінеді. Бұл жағдайда сызықтық адрестен басқа (n, m) сандарының жұбы ретінде берілген виртуалды адрес қолданылады, мұнда n - сегментті, m – сегмент ішінде жылжуды анықтайды (сурет 4.2 б).



Сурет 4.2 - Виртуалды адрестік кеңістіктердің түрлері: жазық (а), сегменттелген (б)

Виртуалды адрестерді физикалық адрестерге түрлендіруді іске асырудың бір-бірінен айырмашылығы өте үлкен екі әдісі бар.

Бірінші жағдайда виртуалды адрестерді физикалық адрестерге ауыстыру бағдарлама жадыға алғаш жүктелген уақытта жүргізіледі. Бағдарлама жүктелінетін физикалық жадының бастапқы адресі, сонымен қатар транслятор берген бағдарламаның адрестік-тәуелді элементтері туралы ақпарат туралы өзінде бар мәліметтер негізінде жүктеуішті жылжытатын - арнайы жүйелік бағдарлама, виртуалды адрестерін физикалық адрестермен алмастыра отырып, бағдарламаны жүктеуді іске асырады

Екінші тәсілдің мағынасы: бағдарлама өзгертілмеген күйде виртуалды адрестерде жадыға жүктелінеді, яғни нұсқау операндтары мен өту адрестері транслятор қалыптастырған мәндерге ие болады. Қарапайым жағдайда, үдерістің виртуалды және физикалық адрестері үзіліссіз біртұтас адрестер облысын құрады, ОЖ виртуалды адрестерді физикалыққа келесі сұлба бойынша іске асырады. Жүктеуде ОЖ виртуалды адрестік кеңістікке қатысты бағдарламалық кодтың нақты орналасуының жылжуын бекітіп алады. Бағдарламаны орындау барысында Ож әрбір қатынаған сайын виртуалды адрес физикалыққа түрлендіріледі.

Соңғы тәсіл ең ыңғайлы тәсіл болып табылады: жылжытушы жүктеуіш бағдарламаны оған бөлінген жадының бір бөлігіне бекітіп қоятын болса, ал виртуалды адрестерді динамикалық түрлендіру бағдарламалық кодты оның барлық орындалу уақытында жылжытуға мүмкіндік береді. Бірақ, жылжытушы жүктеуішті қолдану тиімдірек болады, себебі, бұл кезде виртуалды адресті түрлендіру бір рет қана жүктеу барысында ғана жүргізіледі, ал динамикалық түрлендіруде – бұл адреске әрбір қатынаған сайын түрлендіріледі.

Виртуалды адрестік кеңістік және виртуалды жады – олар әртүрлі механизмдер және олар міндетті түрде ОЖ бір мезетке жүзеге асырылмайды. Үдерістерге арналған виртуалды адрестік кеңістігі бар, бірақ виртуалды адрес механимі жоқ ОЖ көз алдыға елестетуге болады. Бұл сияқты жағдай, егер әрбір үдеріс үшін виртуалды адрестік кеңістіктің көлемі физикалық жадыдан кіші болса, мүмкін болады.

Жалпы виртуалды адрестік кеңістік екі үзіліссіз бөлікке бөлінеді: жүйелік және қолданушылық. Кейбір ОЖ бұл екі бөлімнің (мысалы, Windows NT, OS/2) көлемдері бірдей

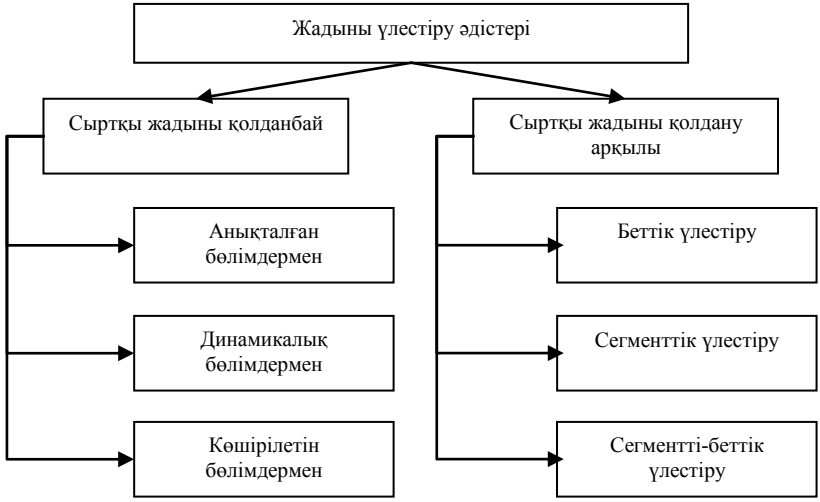
– 2 Гбайттан, бірақ бөлу бөлу түрлі болуы мүмкін, мысалы, 1 Гбайт – ОЖ үшін, 2 Гбайт - қолданбалы бағдарламалар үшін.

Әрбір үдерістің виртуалды адрестік кеңістігінің ОЖ сегменттері үшін бөлінген бөлігі барлық үдерістер үшін бірдей болады. Сондықтан белсенді үдерісті ауыстырған уақытта тек оның жеке сегменттері бар виртуалды адрестік кеңістіктің екінші бөлігі ғана ауыстырылады, көбінде – кодтар және қолданбалы бағдарламаның деректері.

Кез келген типті ОЖ виртуалды жадының жүйелік бөлігіне беттік ығыстыруға ұшырайтын (paged) аймақ және беттік ығыстыру жүрмейтін (non-paged) аймақтар кіреді. Ығыстырылмайтын аймақта ОЖ жылдам реакцияны және/немесе жадының үнемі болуын талап ететін модульдері орналасады, мысалы, ағындар диспетчері немесе жады беттерін ауыстыруды басқаратын код. ОЖ басқа модульдері де қолданушылық сегменттер сияқты беттік ығыстыруға ұшырайды.

4.2.3 Жадыны үлестіру алгоритмдері

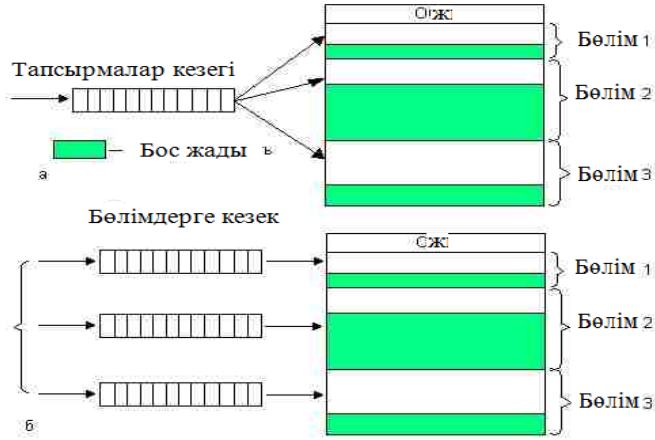
Сурет 4.3 жадыны үлестіру алгоритмдерінің барлығы екі класқа бөлінген: жедел жады мен диск арасында үдеріс сегменттерін жылжытуды қолданатын алгоритмдер және сыртқы жады қолданылмайтын алгоритмдер.



Сурет 4.3 - Жадыны үлестіру әдістерінің жіктелінуі

Жадыны тиянақталған бөлімдермен (фиксированный раздел) үлестіру. Жедел жадыны басқарудың қарапайым әдісінің мәнісі келесідей: жады бөлім деп аталатын тиянақталған (белгілі, шектеулі, фиксированный) көлемді бірнеше облыстарға бөлінеді. Бұл сияқты бөлуді оператор өзі жүйені іске қосқанда немесе оны орнату барысында істей алады. Мұнан кейін бөлімдердің шекаралары ауыспайды.

Орындау үшін жаңа іске қосылған үдеріс жалпы кезекке (сурет 4.4 а) немесе қандайда бір бөлімге кезекке тұрғызылады (сурет 4.4 б).



Сурет 4.4 – Тиянақталған бөлімдер арасында жадыны үлестіру: жалпы кезекпен (а), бөлімдердегі кезектермен (б)

Бұл жағдайда жадыны басқарудың ішкі жүйесі келесі тапсырмаларды шешеді:

- келіп түскен үдеріске қажетті жады көлемін бос бөлімдердің көлемдерімен салыстырып, келетін бөлімді таңдап алады;
- бағдарламаны бөлімдердің біріне жүктеп, адрестердің баптауын жасайды. Бағдарлама құрастырушы трансляциялау кезеңінің өзіне-ақ оны орындайтын бөлімді көрсете алады. Бұл жылжытатын жүктеуішті қолданбай-ақ жадының нақты бір облысына икемделіп бапталған машиналық кодты алуға мүмкіндік береді.

Нақты басымдылығы – жүзеге асурудағы қарапайымдылығында. Бұл әдістің кемшілігі –оның қаттылығы (қатаңдылығы). Себебі, әрбір бөлімде тек бір үдеріс қана орындалады, яғни мильтибағдарламалау деңгейі алдын ала бөлімдер санымен шектелген. Бағдарламаның көлеміне қарамастан ол барлық бөлімді алып тұрады. Мысалы, үш бөлімді жүйеде бір мезетте үштен артық үдерісті қатар орындау мүмкін емес, тіпті оларға қажетті жады көлемі мүлдем аз болса да. Ал бір жағынан, жадыны бөлімдерге бөліп тастау, бөлімдердің ешбіріне сыймайтын, бірақ бірнеше бөлімнің ішіне сыятын үдерістер мен бағдарламаларды орындауға мүмкіндік бермейді.

Жадыны бұл сияқты басқару ертеректегі мильтибағдарламалы ОЖ қолдалынылған. Бірақ, жадыны тиянақталған бөлімдермен үлестіру тәсілі нақты уақыт жүйелерінде қолданыс тауып жатыр.

Жадыны динамикалық бөлімдермен үлестіру. Бұл жағдайда машинаның жадысы алдын ала бөлімдерге бөлінбейді. Алғашында бағдарламалар (қосымшалар) үшін бөлінген жады бос болады. Әрбір орындалуға келіп түскен бағдарлама үшін үдеріс құру барысында оған қажетті жады бөлінеді (егер қажетті жады көлемі болмаса, онда бағдарлама орындалуға алынбайды және оған үдеріс құрылмайды). Үдеріс аяқталысымен оның орнына басқа үдерісті жүктеуге болады. Сонымен, кез келген уақыт мезетінде жедел жады кез келген көлемді бос және бос емес телімдерден (бөлімдерден) тұрады. 4.5 суретте динамикалық үлестіруді қолданғандағы әртүрлі мезеттердегі жадының күйі көрсетілген. t0 мезетінде жадыда тек ОЖ, ал t1 мезетінде жады 5 үдерістің арасында үлестірілген, және П4 үдерісі

аяқталып, жадыдан шыққалы тұр. П4 үдерісінен босаған орынға t3 мезетінде келіп түскен П6 үдерісі жүктелінуде.

Жадыны бұл сияқты әдіспен басқаруды жүзеге асыруға арналған ОЖ функциялары төмендегідей:

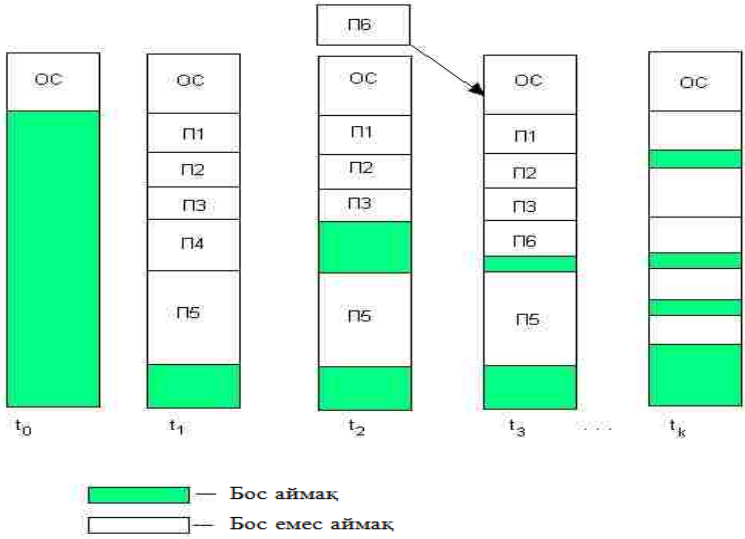
– бөлімдердің бастапқы адрестері мен көлемдері көрсетілетін жиынтық және бос емес облыстардың кестесін жүргізу;

– Жаңа үдерісті құруда — жадыға қойылатын талаптарды талдау, бос облыстар кестесін қарау және бөлімді таңдау;

– Бөлінген бөлімге бағдарламаны жүктеу және бос және бос емес облыстар кестесіне өзгеріс енгізу. Бұл тәсіл бағдарламалық код жылжытылмайды деп есептейді, яғни адрестерді баптауды жүктеумен қатар жүргізуге болады;

Жадыны тиянақталған бөлімдермен (фиксированный раздел) үлестіру әдісімен салыстырғанда бұл әдістің икемділігі (иілгіштігі) жоғарырақ, бірақ бұл әдістің үлкен кемшілігі бар, ол – жадыны фрагментациялау. Фрагментация - өте кіші көлемді (фрагментті) еркін жадыда көршілес емес (сыбайлас емес) телімдердің өте көп болуы. Олардың кішілігі сондай, ешбір келіп түскен бағдарлама олардың ешқайсысына сыймайды, ал бірақ фагменттердің қосынды көлемі қажетті көлемнен едәуір үлкен болуы мүмкін.

Жадыны динамикалық бөлімдермен үлестіру 60-70 жылдардың мультибағдарламалық ОЖ көбінде жадыны басқару ішкі жүйесінің негізінде жатыр, атап айтсақ, аса танымал OS/360 сияқты ОЖ.

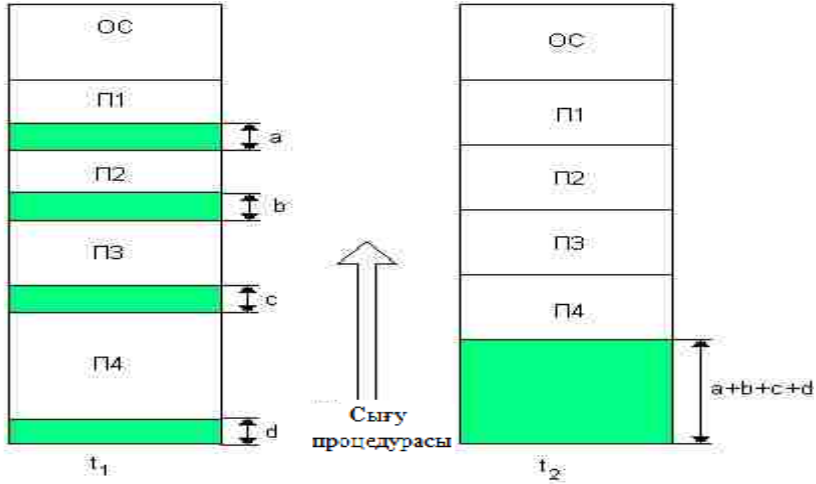


Сурет 4.5 - Жадыны динамикалық бөлімдермен үлестіру

Көшірілетін бөлімдер. Фрагментациямен күресудің әдістерінің бірі бос емес телімдердің барлығын жоғары (үлкен) немесе кіші адрестерге қарай нәтижесінде бос жады

біртұтас бос жады құратындай етіп, жыжыту (сурет 4.6). Бұл жағдайда, ОЖ жадыны динамикалық бөлімдермен үлестіргенде орындайтын функциялармен қатар, бұл жағдайда ОЖ оқтын-оқтын бөлімдердің ішіндегісін жадының бір жерінен екінші жеріне көшірмесін алып, бос және бос емес облыстардың кестесін жөндеп отырады. Бұл процедура сығу деп аталады. Сығу үдеріс аяқталған сайын немесе жаңадан құрылған үдеріс үшін бос бөлім жеткіліксіз болғанда ғана орындалады. Бірінші жағдайда, бос және бос емес облыстар кестесіне өзгеріс енгізуге аз уақыт кетеді, ал екінші жағдайда – сығу процедурасына сирек орындалады.

Сығу тұғырнамасы жадыны үлестірудің басқа әдістері қолданылған мезетте, мысалы әрбір үдеріс үшін жадының біртұтас облысы емес, ал еркін көлемді (сегмент) жадының бірнеше көршілес телімдері берілген уақытта пайдаланылады. Бұл сияқты тәсіл OS/2 ертеректегі нұсқаларында пайдаланылған, ал пайда болған фрагментация сегменттерді жылжыту арқылы жойылып отырған.



Сурет 4.6 - Жадыны көшірілетін бөлімдермен үлестіру

4.2.4 Свопинг және виртуалды жады

Бағдарламаның орындалуының қажетті шарты, оның жедел жадыда орналасуы. Тек бұл жағдайда ғана процессор командаларды алып, оларда көрсетілген әрекеттерді орындай алады. Жедел жады көлемі бір мезетте орындалатын бағдарламалардың санын және олардың виртуалды адрестік кеңістіктерінің көлемін шектейді.

Виртуалды деп қолданушыға немесе қолданушы бағдарламасына шындығында онда жоқ қасиеттері бар ресурс ретінде көрінетін ресурс. Бұл жағдайда қолданбалы бағдарламалаушы қарамағына виртуалды жедел жады беріледі, ал оның көлемі жүйеде бар нақты жедел жады көлемінен үлкен. Жедел жадыны виртуализациялау ОЖ модульдерінің және процессордың аппараттық сұлбасы (схема) жиынтығы арқылы іске асады және келесі тапсырмаларды шешуді қамтиды:

– Түрлі типті есте сақтау құрылғыларға мәліметтерді орналастыру, мысалы, бағдарламалардың кодының бір бөлігін – жедел жадыда, ал бір бөлімін – диске;

– Жедел жадыдан дискке және керісінше жылжыту үшін үдерістердің немесе олардың бөліктерінің бейнелерін таңдау;

– Қажеттітікке байланысты мәліметтерді жады мен диск арасында жылжыту,

– Виртуалды адрестерді физикалыққа түрлендіру.

Дискпен жедел жадыны ортақ қолдануды ұйымдастыру бойынша барлық әрекеттер: жылжытылатын фрагменттер үшін орын бөлу, адрестерді баптау, жүктеу және кері жүктеу үшін кандидаттарды таңдау – бағдарламашының қатысуынсыз, ОЖ мен процессордың аппаратурасымен орындалады және қосымшалардың жұмысының логикасына әсер етпейді.

Жадыны визуализациялау екі әртүрлі тәсілдердің негізінде жүргізілуі мүмкін:

- свопинг (swapping) — үдерістердің свопинг бейнелері дискке жүктелініп, жедел жадыға тұтасымен қайтарылады;
- Виртуалды жады (virtual memory) — жедел жадымен диск арасындағы виртуалды жады үдерістердің бейнелерінің бөліктері (сегменттері, беттер және т.с.с) жылжиды.

Всопинг виртуалды жадының жеке жағдайы, және жедел жадымен дискті ортақ пайдаланудың жүзеге асырылуындағы қарапайымырақ тәсілі. Бірақ, тартқылауға артықшылық тән: ОЖ үдерісті орындау үшін белсендендіруге шешкен кезде, ережеге сай жедел жадыға оның барлық сегменттерін жүктеуді талап етпейді – орындалуға жататын нұсқау бар кодтық сегменттің кішкене ғана бөлігін және бұл нұсқау жұмыс жасайтын деретер сегментінің бөлігін жүктеу жеткілікті, сонымен қатар стек сегменті үшін орын бөлу. Осы сияқты жаңа үдерісті жүктеу үшін жадыны босатқанда көбінесе басқа үдерісті дискке тұтасымен кері жүктеудің қажеттілігі жоқ, дискке тек оның бейнесінің бір бөлігін ғана ығыстырып алу жеткілікті. Артық ақпаратты жылжыту жүйе жұмысын баяулатады, және де жадыны тиімсіз қолдануға әкеледі. Сонымен қатар, свопингті қолдайтын жүйелердің тағы бір маңызды кемшілігі бар: виртуалды адрестік кеңістігінің көлемі бар бос жады көлемінен үлкен үдерісті орындауға жүктеуге қабілетсіз.

Көрсетілген кемшіліктердің әсерінен свопинг жадыны басқарудың негізгі механизмі ретінде заманауи ОЖ мүлдем қолданылмайды. Олардың орнына одан жаңарақ виртуалды жады механизмі келді, айтылғандай жедел жадыда орын жеткіліксіз болған уақытта дискке үдерістердің бейнелерінің тек бөліктері ғана жүктелінеді.

Виртуалды жадының жедел жадыда үдерістердің бейлелерінің немесе олардың бөліктерінің орналасу орнын кө қайтара ауыстырудың нәтижесінде пайда болатын басты мәселесі виртуалды адрестерді физикалық адреске түрлендіру. Бұл мәселенің шешімі жадыны басқарудың жүйесінде виртуалды адрестік кеңістікті құрылымдаудың қандай тәсілі қолданылғанына тәуелді болады. Қазіргі уақытта виртуалды жадыны жүзеге асырудың түрлері үш класпен бейнелеуге болады:

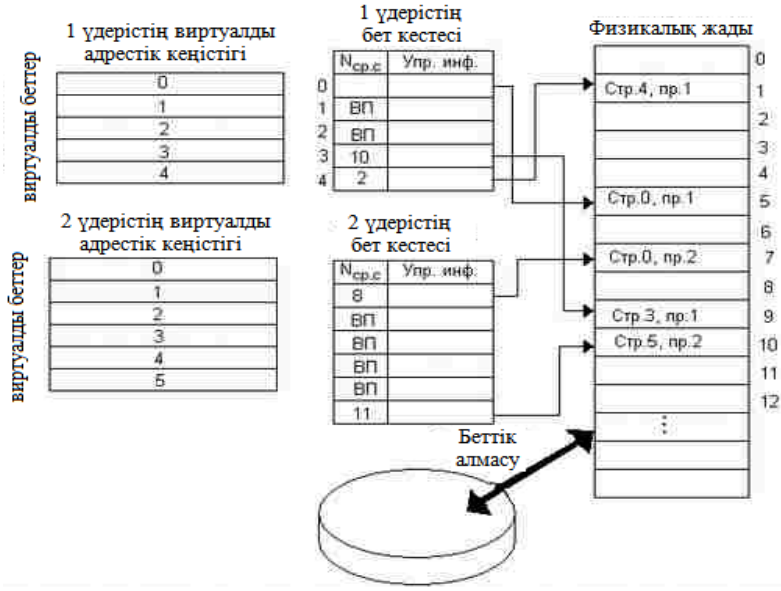
- беттік виртуалды жады деректердің дискпен жады арасында жылжуын бетпен ұйымдастырады – виртуалды адрестік кеңістіктің шектелген көлемді және салыстырмалы түрде шағын көлемді бөліктері;
- сегментті виртуалды жады деректердің сегменттермен жылжуын қарастырады, сегмент –деректердің мағыналық мәнін ескере отырып алынған виртуалды адрестік кеңістіктің кез келген көлемді бөліктері;
- сегментті-беттік виртуалды жады екі деңгейлі бөлуді қолданады: виртуалды адрестік кеңістік сегменттерге бөлінеді, одан кейін сегменттер беттерге бөлінеді. Деректерді жылжытудың өлшем бірлігі бет болады. Жадыны басқарудың бұл тәсілінде жоғарғы екі тәсілдің барлық элементтері бар.

Дискте сегменттер мен беттерді уақытша сақтау үшін көптеген ОЖ дәстүр бойынша свопинг аймағы немесе файлы деп атауды жалғастыратын немесе арнайы аймақ, немесе арнайы файл бөлінеді, дегенмен жедел жады мен диск арасында ақпаратты жылжыту бір үдерісті келесі үдеріспен толық алмастыру формасында емес, бөліктермен жүргізіледі. Бұл

аймақтың кең тараған атауы – беттік файл (page file, или paging file). Беттік файлдың ағымдық көлемі оның маңызды параметрі болып табылады, ол ОЖ мүмкіндіктеріне әсер етеді: беттік файл үлкен болған сайын, ОЖ соғұрлым көп қосымшаларды орындай алады (жедел жады көлемі шектелген (белгілі көлемді) болғанда). Бірақ, бір мезетте қатар жұмыс істейтін қосымшалардың санын беттік файлдың көлемін өсірудің есебінен көбейту олардың жұмысын баяулатады, себебі уақыттың көп бөлігі кодтар мен деректерді жедел жадыдан диске және керісінше жүктеуге кетеді. Заманауи ОЖ беттік файлдың көлемі мультибағдарламалау деңгейі мен жүйенің жылдам әрекет ету арасында ымыраға келу үшін жүйе әкімшісі таңдайтын бапталынатын параметр болып табылады.

Беттік үлестіру. 4.7 суретте жадыны беттік үлестірудің сұлбасы көрсетілген. Әрбір үдерістің виртуалды адрестік кеңістігі виртуалды беттер (virtual pages) деп аталатын бұл жүйе үшін бірдей, анықталған көлемді бөлімдерге бөлінеді. Жалпы жағдайда, виртуалды адрестік кеңістіктің көлемі бет көлеміне еселі, сондықтан әрбір үдерістің соңғы беті жалған аймақпен толтырылады.

Машинаның барлық жедел жадысы физикалық беттер (немесе блоктар, немесе кадрлар) деп аталатын сол сияқты көлемді бөліктерге бөлінеді.бет көлемі екінің дәрежесіне тең етіп таңдалып алынады: 512, 1024, 4096 байт және т.с.с. Бұл адрестерді түрлендірудің механизмдерін қысқартуға мүмкіндік береді.



Сурет 4.7 – Жадыны беттік үлестіру

Үдерісті құру кезінде ОЖ оның бірнеше виртуалды беттерін (кодтық сегменттің және деректер сегментінің бастапқы беттерін) жедел жадыға жүктейді. Үдерістің барлық виртуалды адрестік кеңістігінің көшірмесі дискте тұрады. Көршілес виртуалды беттер міндетті түрде көршілес физикалық беттерде орналаспайды. ОЖ әрбір үдеріс үшін беттер

кестесін құрады – беттер кестесі – үдерістің барлық виртуалды беттері туралы жазбалар бар ақпараттық құрылым.

Кесте жазбасы бет дескрипторы деп аталады, онда келесі ақпараттар болады:

– бұл виртуалды бет жүктелінген физикалық беттің нөмері;

– бар болудың белгісі, егер виртуалды бет жедел жадыда тұрса, бір қойылады;

– бетті өзгерту белгісі, осы бетке қатысты адрес бойынша әр жазу жүргізілген сайын бір қойылады;

– бетке қатынау белгісі, қатынаубиті деп те аталады, осы бетке қатысты адреске әрбір қатынаған сайын бірге қойылады.

Бар болу, өзгерту және қатынау белгілері заманауи процессорлардың көптеген модельдерінде жадымен операцияларды орындағанда процессор сұлбаларымен аппаратты орнатылады. Бет кестелеріндегі ақпарат қандай да бір бетті жады мен диск арасында жылжытудың қажеттілігі туралы сұрақты шешу үшін, сонымен қатар виртуалды адресті физикалыққа түрлендіру үшін қолданылады. Бет кестелерінің өздері деолар сипаттайтын беттер сияқты жедел жадыда орналасады. Бет кестесінің адресі сәйкес үдерістің контекстіне кіреді. Кезекті үдерісті белсендендіру кезінде ОЖ оның бет кестесінің адресін процессордың арнайы регистріне жүктейді.

Жадыға әрбір қатынаған сайын қажетті адрес орналасқан виртуалды беттің адресі ізделінеді, одан кейін ол нөмер бойынша бет кестесінің керекті элементі анықталады, одан бетті сипаттайтын ақпарат алынады. Одан әрі бар болудың белгісі талданады, және егер бұл виртуалды бет жедел жадыда тұрса, онда виртуалды адрестің физикалыққа түрлендірілуі жүргізіледі, яғни виртуалды адрес кестеден көрсетілген физикалық адреспен алмастырылады. Егер көрсетілген виртуалды бет сол уақытта дискке шығарылған болса, онда беттік үзіліс болады. Орындалып жатқан үдеріс күту күйіне ауысады, және дайын күйіндегі кезекте тұрған үдерістерден басқа бір үдеріс белсенді болады. Беттік үзілісті өңдеу бағдарламасы параллельді түрде дисктен керекті виртуалды бетті тауып (ол үшін ОЖ ығыстырылған беттің дисктің беттік файлында орналасуын есте сақтауы керек), және оны жедел жадыға жүктеуге тырысады. Егер жадыда бос физикалық бет болса, онда жүктеу дереу орындалады, ал егер бос беттер болмаса, онда бұл жүйеде қабылданған беттерді ауыстыру стратегиясы негізінде қай бетті жедел жадыдан шығарып алу алған жөн болатындығы туралы сұрақтың шешімі қарастырылады.

Сегменттік үлестіру. Үдерістің виртуалды адрестік кеңістігі бөліктерге – көлемі онда орналасқан ақпараттың мағыналық мәнін ескере отырып, анықталатын сегменттерге бөлінеді. Жеке сегменттің өзі кіші бағдарлама, деректер массиві және т.с.с. болуы мүмкін. Виртуалды адрестік кеңістікті сегменттерге бөлуді компилятор бағдарламашының нұсқауы немесе жүйеде қабылданған келісімге сәйкес бастапқы берілгендей бойынша жүзеге асырады. Сегменттің максималды көлемі виртуалды адрестің разрядтылығымен анықталады, мысалы, процессордың 32-разрядтық ұйымдастырылуанды ол 4 Гбайтқа тең. Бұлай болғанда үдерістің максималды мүмкін болатын виртуалды адрестік кеңістігі әрқайсысы 4 Гбайт болатын N вмртуалды сегмент болады. Сегменттер бір-біріне қатысты реттелмейді, сондықтан сегменттерге ортақ сызықтық виртуалды адрестер жоқ, виртуалды адрес сандар жұбымен беріледі: сегмент нөмері және сегмент ішіндегі сызықтық виртуалды адрес.

Реентерабельділік (reentrantable) — кодтың қайта кіру қасиеті, ол бір мезетте оны бірнеше үдерістің қолдануына мүмкіндік береді. Реентерабельді кодты орындағанда үдерістер оны өзгертпейді, сондықтанда жадыға кодтың көшірмесінің бір данасын ғана жүктеу жеткілікті болады.

Үдерісті жүктегенде оның сегменттерінің бір бөлігі ғана жедел жадыға орналастырылады, виртуалды адрестік кеңістіктің толық көшірмесі дисктің жадыда болады. Әрбір жүктелінген сегмент үшін ОЖ жеткілікті көлемді бос жадының үзіліссіз аумағын іздейді. Бір үдерістің көршілес виртуалды жады сегменттері жедел жадыда көршілес

бөліктерді алуы мүмкін. Егер үдерісті орындау кезінде сол мезетте жадыда жоқ сегментке қатысты виртуалды адрес бойынша қатынау болса, онда үзіліс орын алады. ОЖ белсенді үдерісті тоқтатып қояды, кезектен келесі үдерісті орындауға жібереді, ал параллельді түрде дискен керекті сегментті жүктеуді ұйымдастырады. Жадыда сегментті жүктеу үшін орын болмаған жағдайда, ОЖ шығарып алу үшін сегментті таңдайды, бұл жерде жадыны басқарудың беттік тәсіліндегі бетті таңдаудағы критерилері сияқты критерилерді қолданады.

Үдерісті құру кезеңінде оның бейнесін жедел жадыға жүктеу кезінде жүйе үдерістің сегменттер кестесін құрады (беттер кестесі сияқты), онда әрбір сегмент үшін:

- сегменттің жедел жадыдағы физикалық адресі;
- сегмент көлемі;
- сегментке қатынау ережелері;
- өзгерту, бар болу және бұл сегментке қатынау белгісі, және тағы басқа ақпарат.

Егер бірнеше үдерістің виртуалды адрестік кеңістіктеріне бір сегмент кіріп тұрса, онда бұл үдерістердің сегменттер кестесінде бұл сегмент бір данамен жүктелінетін жедел жадының бір бөлігіне сілтеме жасалынады.

Сегменттік үлестірудің кемшілігі оның артықшылығы. Сегменттік ұйымдастыруда жады мен диск арасында жылжытудың бірлігі сегмент болады, ал жалпы жағдайда ол беттен үлкен көлемді алады. Бірақ, көптеген жағдайларда бағдарламаның жұмысы үшін сегментті тұтасымен жүктеудің қажеті жоқ, бір немесе екі беттің өзі-ақ жеткілікті болады. Жоғарыдағыдай жадыда бос орын болмағанда тұтас сегментті жығарып алудың қажеті жоқ, бірнеше бетті ғана шығарып алуға болады.

Бірақ, сегменттік үлестірудің басты кемшілігі – бұл сегменттер көлемдерінің болжап болынбайтындығынан туындайтын фрагменттау. Жүйе жұмысының барысында бірде-бір сегмент жүктелінбейтін шағын бос жады аумақтары пайда болады. Бұл фрагменттер алатын қосынды көлем жүйенің жалпы көлемінің бірталай бөлігін алуы мүмкін, тиісінше жадыны тиімсіз қолдануға әкеледі.

Жадыны сегменттік ұйымдастырудың беттік ұйымдастырудан үлкен айырмашылығының бірі үдерістің сегменттеріне оған қатынаудың дифференциалданған құқықтарын беру мүмкіншілігі. Мысалы, қосымша үшін керекті бастапқы ақпарат тұрған бір деректер сегментінде «тек қана оқу үшін» құқығы, ал нәтижелерді бейнелейтін деректер сегментінде «оқу және жазу» құқығы болуы мүмкін. Бұл қасиет жадының сегменттік моделіне беттіктің алдында үлкен басымдылық береді.

Сегменттік-беттік үлестіру. Бұл әдіс жадыны басқарудың беттік және сегменттік механизмдерінің комбинациясы болып табылады, және екі әдістің де құндылықтарын жүзеге асыруға бағытталған.

Жадыны сегменттік ұйымдастырудағы сияқты үдерістің виртуалды адрестік кеңістігі сегменттерге бөлінген. Бұл бағдарламаның кодтары мен деректерінің әртүрлі бөліктеріне әртүрлі құқықтар анықтауға мүмкіндік береді.

Жады мен диск арасында деректерді жылжыту сегменттермен емес, беттермен жүргізіледі. Ол үшін әрбір виртуалды сегмент және физикалық жады бірдей өлемді беттерге бөлінеді, ал бұл болса фрагменттауды минимумға жеткізіп, жадыны тиімдірек пайдалануға мүмкіндік береді.

4.2.5 Деректерді кэштеу

Кэш-жады немесе кэш (cache), — бұл бір-бірінен қатынау уақыты және жиі қолданылатын ақпаратты «баяу» ЕСҚ-нан «Жылдам» ЕСҚ-на динамикалық көшірмелеудің есебінен деректерді сақтау құны бойынша айырмашылықтары бар екі типті есте сақтау құрылғыларының бірлесе қызмет етуі, бұл бір жағынан деректерге қатынаудың орташа уақытын азайтуға, ал екінші жағынан қымбатырақ жылдам жұмыс істейтін жадыны үнемдейді.

Кэш-жады немесе кэш деп тек қана екі типті есте сақтау құрылғыларын ұйымдастыруды ғана емес, есте сақтау құрылғыларының бірі – «жылдам» ЕСҚ атайды.

Кэштеу – бұл жедел жадыға, диске және басқа да есте сақтау құрылғыларына қатынауды жылдамдатуға жарамды бірегей әдіс. Егер кэштеу жедел жадыға қатынаудың орташа уақытын азайту үшін қолданылатын болса, онда кэш ретінде жылдам жұмыс істейтін ститикалық жады қолданылады. Егер кэштеу дискте сақталған деректерге қатынауды жылдамдату үшін енгізу-шығару жүйесімен қолданылатын болса, онда кэш-жадының рөлін жедел жадыдағы белсенді қолданылатын деректер тұрып қалатын буферлер ойнайды. Виртуалды жадыны да деректерді кэштеудің қадиғатының жүзеге асырылуының бір нұсқасы деп санауға болады, бұл жерде жедел жады сыртқы жадыға – қатты дискке қатысты кэштің рөлін ойнайды. Бұл жағдайда кэштеу деректерге қатынау уақытын азайту үшін емес, белсенді үдерістерге орын босату мақсатында қолданылмайтын код және деректерді дискке уақытша жылжыту есебінен дискке жартылай жедел жадыны алмастырту үшін қолданылады

4.3 Бақылау сұрақтары

4.3.1 Жадыны басару бойынша ОЖ функциялары қандай?
4.3.2 Виртуалды адрес физикалық адрестен қандай белгілерімен өзгеше?
4.3.3 Виртуалды адрестерді физикалық адрестерге түрлендіру қалай жүргізіледі?
4.3.4 Жадыны үлестіру алгоритмдері қандай кластарға бөлінеді?
4.3.5 Жадыны анықталған бөлімдермен үлестіру қалай жүргізіледі?
4.3.6 Жадыны динамикалық бөлімдермен үлестіру қалай жүргізіледі?
4.3.7 Жадыны көшірілетін бөлімдермен үлестіру қалай жүргізіледі?
4.3.8 Жадыны беттік үлестіру қалай жүргізіледі?
4.3.9 Жадыны сегменттік және сегменттік-беттік үлестіру қалай жүргізіледі?
4.3.10 Кэш-жады және кэштеу дегеніміз не?

5 ЕНГІЗУ-ШЫҒАРУДЫ БАСҚАРУ

5.1 Дәріс мақсаты

Дәріс мақсаты енгізу-шығаруды басқару бойынша ОЖ міндеттерімен және енгізу-шығарудың базалық ішкі жүйесінің функцияларымен танысу болып табылады.

5.2 Теориялық мәліметтер

ОЖ негізгі функцияларының бірі компьютердің барлық енгізу-шығару құрылғыларын басқару болып табылады. ОЖ құрылғыларға командаларды жіберуі, үзілулерді қағып алуы және қателерді өңдеуі қажет, сол сияқты құрылғылардың арасында және жүйенің қалған бөліктерінің арасында интерфейсті қамтамасыз етуі қажет. Дамыту мақсатында құрылғылардың барлық түрі үшін интерфейс біркелкі болуы қажет (құрылғылардан тәуелсіздік).

5.2.1 Енгізу-шығару құрылғысының физикалық ұйымдастырылуы

Енгізу-шығару құрылғысы екі түрге бөлінеді: блок-бағдарланған құрылғы және байт-бағдарланған құрылғы. Блок-бағдарланған құрылғы блогында әрқайсысының өзінің меншікті адресі бар, тиянақталған өлшемді ақпараттарды сақтайды. Блок-бағдарланған құрылғының ең көп таралған түрі- диск. Байт-бағдарланған құрылғы адрестелмеген және іздеу операциясын жүргізуге мүмкіндік бермейді, олар байттың реттілігін пайдаланады және генерациялайды. Терминал, жолдық принтерлер, желілік адаптерлер осының мысалы бола алады.

Жалпы, сыртқы құрылғы механикалық және электрондық құрауыштардан тұрады. Электрондық құрауыш құрылғы контроллері немесе адаптер деп аталады. Құрылғының өзі механикалық құрауыш болып табылады. Кейбір контроллерлер бірнеше құрылғыны басқаруы мүмкін. Егер контроллер мен құрылғының арасындағы интерфейс қалыпталған (стандарт болса, онда тәуелсіз өндірушілер сәйкес контроллерларды қалай шығарса, құрылғыларды да солай шығара алады.

Жалпы, операциялық жүйе құрылғымен емес, контроллермен жұмыс істейді. Контроллер, ереже бойынша, қарапайым функцияларды орындайды, мысалы, байттан тұратын бит ағынын блокқа түрлендіріеді, бақылауды жүзеге асырады және қателерді түзетеді. Әрбір контроллерде орталық процессорымен өзара әрекет үшін қолданылатын бірнеше регистр болады. Кейбір компьютерлерде бұл регистрлер физикалық адрестік кеңістіктің бөлігі болып табылады. Мұндай компьютерлерде енгізу – шығару операциялары жоқ. Басқа компьютерлерде көбінесе порт деп аталатын енгізу – шығару регистрлерінің адресі, арнайы енгізу – шығару операцияларын кіргізу есебінен,меншікті адрестік кеңістік түзеді.

ОЖ контроллер регистріне команданы жаза отырып, енгізу - шығаруды орындайды. Мысалы: IBM PC иілмелі диск контроллері мына сияқты READ, WRITE, FORMAT және тағы басқа 15 команданы қабылдайды. Қашан команда қабылданады, сол кезде процессор контроллерді қалдырады және басқа жұмыспен айналысады. Команда аяқталғанннан кейін операция нәтижесін тексеруі тиіс операциялық жүйенің процессорына басқаруды беру үшін, контроллер үзілу ұйымдастырады. Процессор контроллер регистрінен ақпаратты оқи отырып, нәтижесін ала отырып, құрылғы статусына ие болады.

5.2.2. Енгізу – шығару бағдарламалық жабдықтауды ұйымдастыру.

Енгізу – шығаруды бағдарламалық жабдықтауды ұйымдастырудың негізгі идеясы оны бірнеше деңгейге бөлуде болып тұр, төменгі деңгейлер жоғарғы деңгейдегі аппаратура

ерекшеліктерінің экрандалуын қамтамасыз етеді, ал олар өз кезегінде, қолданушылар үшін ыңғайлы интерфейс қамтамасыз етеді

Құрылғыларға тәуелсіздік өзекті мәселе болып табылады. Бағдарлама түрі ол мәліметтерді алмалы дискіден немесе қатты дискіден оқыды ма оған байланысты болмауы қажет.

Енгізу- шығаруды бағдарламалық жабдықтау үшін, қателерді өңдеу басқа маңызды сұрақ болып табылады. Жалпы айтқанда, қателерді мүмкіндігінше аппаратураға жақынырақ өңдеу қажет. Егер контроллер оқуда қате барын байқаса, онда ол түзетуге тырысуы қажет. Егер де оған бұл мүмкін болмаса. онда қателерді түзетумен құрылғы драйвері айналысуы қажет. Көптеген қателер енгізу – шығару операцияларын орындауды қайталауға ұмтылғанда жоғалуы мүмкін, мысалы: дискідегі немесе оқу құралының ұшында шаң-тозаңның болуымен байланысты. Егер де төменгі деңгей қатені түзете алмаса, ол осы қате туралы жоғары деңгейге хабарлайды.

Тағы бір өзекті мәселе – бұл бұғатталатын (синхронды) және бұғатталмайтын (асинхронды) жіберулерді пайдалану. Физикалық енгізу – шығару операцияларының көпшілігі асинхронды орындалады – процессор жіберуді бастайды және үзілу басталғанға дейін, басқа жұмысқа ауысады. Егер енгізу – шығару операциялары, бұғатталатын - READ командасынан кейбір мәліметтер бағдарлама буферіне түскенге дейін, автоматты түрде тоқтай тұратын болса, сонда қолданушылықты бағдарламаларды жазу біршама жеңілірек болады. ОЖ енгізу – шығару операцияларын асинхронды орындайды, бірақ оларды қолданушылық бағдарлама үшін синхронды түрде көрсетеді.

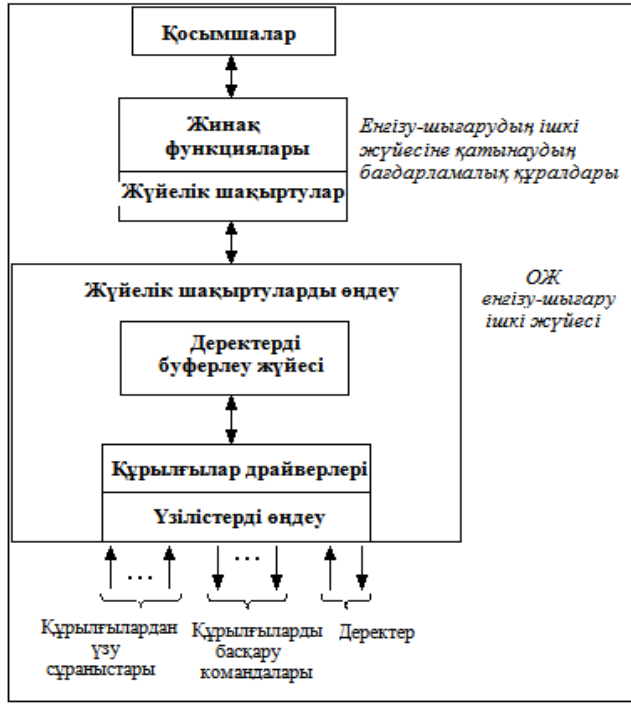
Соңғы мәселе, бір құрылғылар бөлінісілетін, ал кейбіреулері ерекшеленген болып келетіндігінде тұр. Диск - бұл бөлінісілетін құрылғы, өйткені бір мезгілде бірнеше қолданушы дискіге қатынас құруы ешқандай мәселе тудырмайды. Принтер – бұл ерекшеленген құрылғы, өйткені әртүрлі қолданушылар басып шығарған жолдарды араластыруға болмайды. Ерекшеленген құрылғының болуы операциялық жүйе үшін қандай да бір мәселелер тудырады.

Бұл қойылған мәселені мақсатты шешу үшін, енгізу – шығару бағдарламалық қамтуды 4 қабатқа бөлу керек (9-сурет):

- үзілулерді өңдеу,
- құрылғы драйверлері,
- құрылғыларға тәуелсіз операциялық жүйе қабаты,
- бағдарламалық жабдықтауды пайдалану қабаты.

Үзілулерді өңдеу. Мүмкіндігінше операциялық жүйенің неғұрлым аз бөлігі үзілулермен байланысты болуы үшін, оларды мүмкіндігінше операциялық жүйе қойнауында тереңірек жасыру қажет. Енгізу-шығару операциясын инициализациялайтын үдеріске, үзілу басталғанға дейін және операция аяқталғанға дейін, өзін бұғаттау мүмкіндігін беруді шешу неғұрлым жақсы әдіс болып табылады.

Үдеріс мынадай шақыруларды пайдалана отырып, өзін бұғаттай алады , мысалы: DOWN шақыруы семафор үшін, немесе WAIT шарт айнымалысы үшін, немесе RECEIVE хабарламаларды күту үшін. Үзілу басталғанда, үзілулерді өңдеу процедурасы UP, SIGNAL шақыруларын немесе үдеріске хабарлама жіберуді пайдалана отырып, енгізу-шығару операциясын инициализациялайтын үдерісті бұғаттаудан босатады. Кез келген жағдайда, үзілуден келетін тиімділік мынада болып табылады: бұрынырақта бұғатталған үдеріс, енді өзінің орындауын жалғастырады.



Сурет 5.1 - Енгізу – шығару ішкі жүйесін көпдеңгейлі ұйымдастыру

Құрылғы драйвері. Құрылғыға байланысты барлық код құрылғы драйверіне орналастырылады. Әрбір драйвер біртүрлі немесе бірқласты құрылғыларды басқаруы мүмкін.

Операциялық жүйеде тек қана құрылғы драйвері ғана, қандай да бір құрылғының нақты ерекшеліктері жайлы біледі. Мысалы: тек диск драйвері ғана жолмен, секторлармен, жазу құрылғысының ұшын орнату уақытымен және дискінің дұрыс жұмыс істеуін қамтамасыз ететін басқа да факторлармен жұмыс істейді .

Құрылғы драйвері бағдарламалық қабат құрылғысынан сұраныс қабылдайды және оны қалай орындау керектігін шешеді. Мәліметтердің n блогын оқу осы типтес сұраныс болып табылады. Егер драйвер сұраныстың келіп түсу уақытында бос болса, онда ол бірден сұранысты орындауды бастайды. Егер де ол басқа сұранысқа қызмет көрсетумен айналысып жатса, онда қайтадан келіп түскен сұраныс, бұрынғы бар сұраныстардың кезегіне тіркеледі және ол қашан өз кезегі келеді, сонда орындалатын болады.

Енгізу – шығару сұранысын іске асырудағы бірінші қадам, мысалы, диск үшін оны абстрактілі түрден нақтыға түрлендіруде болып отыр. Дискілік драйвер үшін, бұл блок нөмірлерін цилиндрлердің, ұштардың, секторлардың нөмірлеріне түрлендіру, мотордың жұмыс істеуін, ұш керекті цилиндрдің үстінде тұруын тексеру болып табылады. Қысқаша айтқанда, ол контроллердің қандай операцияларды және қандай реттілікпен орындайтынын шешуі қажет.

Драйвер команданы контроллерге бергеннен кейін, ол өзін берілген операция аяқталғанға дейін бұғаттауы керек пе, жоқ па екендігін шешуі керек. Егер операция, мәліметтердің қандай да бір блогын басып шығарғандағы сияқты, елеулі уақыт алатын болса, онда драйвер операция аяқталғанға дейін бұғатталады және үзілуді өңдеуші оны бұғатталудан босатпайды. Егер енгізу – шығару командасы жылдам орындалса, (мысалы: экранды айналдыру), онда драйвер оның аяқталуын бұғатталмай тосады.

Операциялық жүйенің құрылғыға тәуелсіз қабаты. Енгізу – шығаруды бағдарламалық жабдықтаудың көбірек бөлігі құрылғыға тәуелсіз болып табылады. Драйверлердің өзара арасындағы және құрылғыға тәуелсіз бағдарламалардың арасындағы дәл шекара жүйемен анықталады, өйткені тәуелсіз әдістермен іске асыруға болатын кейбір фцункциялар, шын мәнінде, тиімділігін жоғарылату үшін немесе басқа қандайда бір себептерге байланысты драйверлер түрінде орындалған.

Құрылғыға тәуелсіз қабат үшін тән функциялар мыналар болып табылады:

- құрылғы драйверін жалпы интерфейспен қамтамасыз ету;
- құрылғыға атау беру;
- құрылғыны қорғау;
- блокты тәуелсіз өлшеммен қамтамасыз ету;
- буферлеу;
- блок – бағдарланған жүйедегі жадыны үлестіру;
- ерекшеленген құрылғыларды үлестіру және босату;
- қателер туралы хабардар ету.

Берілген тізімдегі кейбір функцияларға тоқталамыз.

Бағдарламалық жабдықтаудың жоғарғы қабатына әр түрлі өлшемді блокпен жұмыс істеу ыңғайсыз, сондықтан бұл қабат блокты бірыңғай өлшеммен қамтамасыз етеді. Мысалы: бірнеше әр түрлі блоктарды бір логикалық блокқа біріктіру есебінен іске асырады. Осыған байланысты жоғарғы деңгейлер, физикалық сектордың өлшеміне тәуелсіз логикалық блоктың бірыңғай өлшемін пайдаланатын, абстрактілік құрылғылармен ғана жұмыс істейді.

Файлды құруда немесе оны жаңа мәліметтермен толтыруда, оған жаңа блоктар бөлу қажет. Бұл үшін ОЖ тізім енгізуі немесе дискінің бос блоктарының биттік картасын енгізуі қажет. Дискіде бос орын бар екендігі туралы ақпараттың негізінде, драйвер қабатынан жоғары орналасқан, құрылғыға тәуелсіз және бағдарламалық қабат іске асыратын бос блокты іздеу алгоритмін қарастыру мүмкін болады.

Бағдарламалық жабдықтаудың қолданушылықты қабаты. Енгізу – шығару бағдарламалық жабдықтаудың көбірек бөлігі ОЖ ішінде орналасқанымен, оның кейбір бөліктері қолданушылықты бағдарламалармен байланыстыратын кітапханаларда болып табылады. Жалпы енгізу – шығару шақырулары кіретін, жүйелік шақырулар, әдетте кітапхана процедураларымен жасалады.

Егер С тілінде жазылған бағдарламада, const=write (fd, buffer, nbytes) шақырулары бар болса, онда write кітапханалық процедурасы бағдарламамен байланысты болады. Осыған ұқсас процедуралар жиыны енгізу – шығару жүйесінің бөлігі болып табылады. Көбінесе, енгізу немесе шығаруды форматтау кітапхана прцедураларымен орындалады. Бұған С тілінде, формат жолын және кейбір айнымалыларды енген ақпарат ретінде қабылдайтын, сонан кейін ASCII символдарының тіркесін тұрғызатын және осы тіркесті шығару үшін write шақыруын жасайтын, write функциясы мысал бола лады. Енгізу – шығарудың стандартты кітапханасы, енгізу–шығаруды орындайтын және қолданушылық бағдарламасының бөлігі болып табылатын көптеген процедуралардан тұрады.

Спулинг ішкі жүйесі енгізу – шығару бағдарламалық – жабдықтауының басқа санаты болып табылады. Спулинг – бұл мультибағдарламалық жүйеде ерекшеленген құрылғылармен жұмыс істеу тәсілі.

Осы типтес құрылғыны қарастырамыз, бұл спулинг талап ететін – жолдық принтер

болып табылады. Техникалық жағынан әрбір қолданушылық үүдерісіне принтермен байланысты арнайы файл ашу жеңіл болғанымен, мұндай тәсіл, қолдану үдерісі принтерді кез келген таңдалған уақытқа монополияға алуы мүмкін болғандықтан, қауіпті болып табылады. Мұның орнына, арнайы үдеріс – монитор құрылады, осы үдеріс осы құрылғыны пайдалану құқын алады. Сол сияқты, спулинг каталогы -арнайы каталог құрылады. Файлды басып шығару үшін пайдалану үдерісі шығаратын ақпаратты осы файлға орналастырады. Монитор- үдеріс спулинг каталогындағы барлық файлдарды кезек бойынша басып шығарады

5.3 Бақылау сұрақтары

5.3.1 Енгізу-шығару физикалық ұйымдастырылуын сипаттаңыздар.

5.3.2 Енгізу-шығару бағдарламалық қамтамасының ұйымдастырылуын сипаттаңыздар.

5.3.3 Енгізу-шығару бағдарламалық қамтамасы қандай қабаттарға бөлінеді?

5.3.4 Енгізу-шығару бағдарламалық қамтамасының әрбір қабатын сипаттаңыздар.

6 ФАЙЛДАРДЫ БАСҚАРУ. ФАЙЛДЫҚ ЖҮЙЕЛЕР

6.1 Дәріс мақсаты

Дәріс мақсаты файлдардың логикалық және физикалық ұйымдастырылуы, сол сияқты файлдық жүйелермен (FAT, NTFS) таныстыру.

6.2 Теориялық мәліметтер

Файлдық жүйе - міндеті қолданушыны дискіде сақталған мәліметтермен жұмыс істеуіне ыңғайлы интерфейспен қамтамасыз ету және бірнеше қолданушының файлдарды бірігіп пайдалануын қамтамасыз ету болып табылатын операциялық жүйенің бөлігі.

Кең мағынасында алғанда «файлдық жүйе» дегеніміз:

- дискідегі барлық файлдардың жиынтығы;
- файлдарды басқару үшін қолданылатын мәліметтер құрылымының жиыны, мысалы, мынадай: файлдар катологы, файлдар дескрипторлары, дискідегі бос емес және бос кеңістіктің бөліну кестесі;
- файлдарды басқаруды жүзеге асыратын жүйелік бағдарлама құралдарының кешені, файлдармен орындалатын құру, жою, оқу, жазу, белгілеу, іздеу және басқа операциялар.

Файлдық жүйе:

- файлдардың символдық атауларын ішкі идентификаторлармен белгілейді;
- қолданушы немесе қолданбалы бағдарлама жұмыс істейтін файлдардың символдық атауларын немесе ішкі идентификаторларын дискідегі және басқа жинақтауыштардағы физикалық адрестерге түрлендіреді;
- файлдарға бірлескен қатынас құруды ұйымдастырады;
- файлдарды рұқсат етілмеген қатынас құрудан қорғайды.

Файлдық жүйенің қызмет етуі, файлдық жүйенің сұранысы бойынша, мәліметтерді оперативтік жады мен сыртқы жинақтауыштардың арасында беруді жүзеге асыратын, сыртқы құрылғыларды басқарудың ішкі жүйесінің қызмет етуімен тығыз байланысты

Файлдардың аттары. Файлдар атаулармен идентификацияланады. Қолданушылар файлдарға символдық атаулар береді, мұнда ОЖ пайдаланылатын символдарға және сол сияқты атаудың ұзындығына қойылатын шектеулері есепке алынады.

Біраз уақыт бұрынғыға дейін бұл шектеулер өте тар көлемде болды. Танымал FAT файлдық жүйесінде атау ұзындығы белгілі 8.3 сызбасымен шектеледі (меншікті ат – 8 таңба, кеңейтілімі – 3 таңба), ал ОС UNIX System V атауда 14 таңбадан жоғары болмайды. Бірақ та, қолданушыға ұзын аттармен жұмыс істеу неғұрлым ыңғайлы, өйткені бұл файлда не барын жеткілікті үлкен уақыт аралығы өтсе де, еске түсіруге мүмкіндік беретін атау беруге мүмкіндік туады. Сондықтан, қазіргі заманғы файлдық жүйелер, ереже бойынша , файлдарға ұзын символдық атаулар берілуін қолдайды. Мысалы: Windows NT, өзінің NTFS жаңа файлдық жүйесінде, аяқтайтын нөлдік символды санамағанда, файл атауында 255 символға дейін болуына болатындығын тағайындаған .

Әдетте, әр түрлі файлдар бірдей символдық атауларда болуы мүмкін. Бұл жағдайда, файл катологтардың символдық атауларының реттілігін көрсететін, құрама атаумен бірмәнді идентификацияланады. Кейбір жүйелерде сол бір файлға бірнеше әр түрлі атау беруге болмайды, ал басқа бір жүйелерде мұндай шектеулер болмайды. Соңғы жағдайда, операциялық жүйе файлға қосымша бірегей ат береді, сөйтіп оның бірегей аты мен файлдың арасында өзара бірмәнді сәйкестік орнатуға болады.

Бірегей атау сандық идентификатор болып табылады және операциялық жүйе бағдарламаларымен қолданылады.

UNIX жүйесіндегі индекстік дескриптордың нөмірі файлдың бірегей атының мысалы болып табылады.

Файлдардың түрлері. Файлдар әртүрлі болады: қарапайым файлдар, арнайы файлдар, католог – файлдар. Мәтіндік файлдар, ASCII кодында көрсетілген, символ тіркестерінен тұрады. Бұл құжаттар, бағдарламаның негізгі мәтіні және т.б. болуы мүмкін. Мәтіндік файлдарды экраннан оқуға және принтерде басып шығаруға болады.Екілік файлдар ASCII кодтарын пайдаланбайды,өйткені көбінесе олардың ішкі құрылымдары күрделі болады, мысалы, бағдарламаның объектілік коды немесе мұрағаттық файл. Барлық операциялық жүйелер тіпті болмағанда өздерінің меншікті орындайтын файлы болып табылатын, файлдардың бір түрін таңи білуі керек (должны уметь).

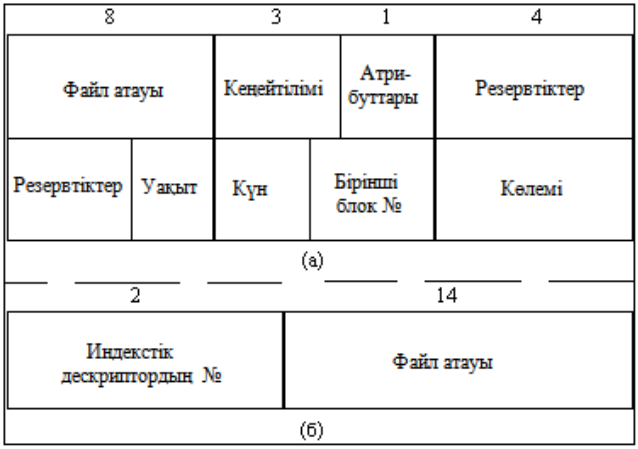
Арнайы файлдар – бұл, файлға жазу немесе файлдан оқу қарапайым командасын пайдаланып, қолданушыға енгізу – шығару операцияларын орындауға мүмкіндік туғызатын, енгізу – шығару құрылғыларымен орайластырылған файлдар. Бұл командалар алдымен файлдық жүйенің бағдарламаларымен өңделеді, ал содан кейін сұранысты орындаудың қандай да бір кезеңінде сәйкес құрылғыны басқару командасына түрленеді. Арнайы файлдар, сол сияқты енгізу – шығару құрылғылары да, блок – бағдарланған және байт бағдарланған болып бөлінеді.

Католог – бұл бір жағынан қолданушының қандай да бір пайымдауымен біріктірілген файлдар тобы (мысалы: ойын бағдарламасы жазылған файлдар немесе бір бағдарламалық пакет құратын файлдар), ал келесі жағынан – бұл оны құрайтын файлдар тобы туралы жүйелік ақпараты бар файлдар.Катологта оған кіретін файлдардың тізімі болады және файлдар мен олардың сипаттамаларының (атрибуттарының) арасындағы сәйкестілік тағайындалады.

Әр түрлі файлдық жүйелерде атрибут ретінде әртүрлі сипаттамалар пайдаланылуы мүмкін. Мысалы:

- рұқсат етілген қатынас құру туралы ақпарат;
- файлға қатынас құру үшін құпия сөз;
- файл иесі;
- файл құрушысы;
- « оқу үшін ғана» белгісі;
- «жасырынған файл» белгісі;
- «жүйелік файл» белгісі;
- «мұрағаттық файл» белгісі;
- «екілік/символдық»белгісі;
- «уақытша» белгісі (үдеріс аяқталғаннан кейін жойылады);
- бұғаттау белгісі;
- жазба ұзындығы;
- жазбадағы түйінді сөзге нұсқағыш;
- кілт ұзындығы;
- соңғы қатынас құру және соңғы өзгерту жасалған уақыт;
- файлдың ағымдағы өлшемі;
- файлдың максималды өлшемі.

Каталогта, тікелей MS - DOS файлдық жүйесінде жасалған сияқты, файл сипаттамалары болуы мүмкін немесе осылардың UNIX ОЖ (сурет 6.1) іске асырылғаны сияқты, осы сипаттамалар жинақталған кестеге жүгінеді. Неғұрлым төменгі деңгейлі каталог, неғұрлым жоғары деңгейлі каталогқа кіруі есебінен, қаталогтар иерархиялық құрылым түзуі мүмкін (сурет 6.2).



Сурет 6.1 - Каталогтар құрылымы: а - MS-DOS қаталогының жазылу құрылымы (32 байт); б - ОС UNIX ОЖ қаталогының жазылу құрылымы

Каталогтар иерархиясы ағаш немесе желі болуы мүмкін. Егер файлға бір ғана каталогқа кіруге рұқсат етілген болса, каталог ағаш (бұтақ) түзеді, ал егер файл бірден бірнеше каталогқа кіретін болса, онда желі түзеді. MS - DOS- та каталогтар ағаш түріндегі құрылым түзеді, ал UNIX те – желілік.

Кез келген басқа файлдар сияқты, каталогтың символдық атауы болады және түбіреінен берілген каталогқа дейінгі жолдан өтетін, барлық каталогтардың символдық атауларынан тұратын тізбектен құралған, құрама атаулармен бір мәнді идентификацияланады.

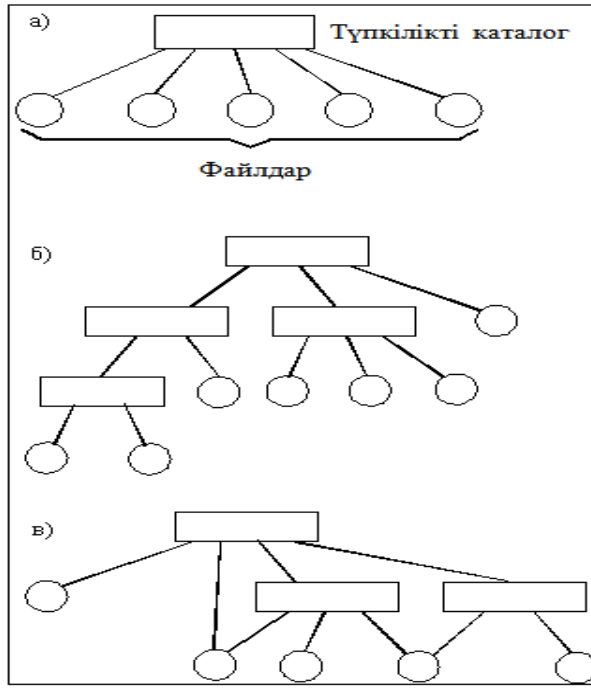
6.2.1. Файлдардың логикалық ұйымдастырылуы

Бағдарламашы файлды логикалық ұйымдастырылған жазба түрінде көрсете отырып, файлдың логикалық ұйымдастырылуымен жұмыс істейді.

Логикалық жазба – бұл бағдарламашы сыртқы құрылғылармен алмасуда операциялайтын мәліметтердің ең кішірек элементі.

Егер тіпті, құрылғымен физикалық алмасу үлкен бірліктермен жүзеге асырылатын болса да, операциялық жүйе бағдарламаның жеке логикалық жазбаларға қатынас құруын қамтамасыз етеді. 6.3 суретте файлды логикалық ұйымдастырудың бірнеше сызбасы көрсетілген.

Жазбалар тиянақталған ұзындықта немесе айнымалы ұзындықта болуы мүмкін. Жазбалар файлда ретті (ретті ұйымдастыру) орналасуы мүмкін немесе жеке логикалық жазбаларға жылдам қатынас құруды қамтамасыз етуді мүмкін ететін, индекстік кесте деп аталатын, кестені пайдаланып, неғұрлым күрделі тәртіппен орналасуы мүмкін (индекс – ретті ұйымдастыру) . Жазбаны идентификациялау үшін кілт деп аталатын арнайы жазба жазықтығы қолданылуы мүмкін. ОС UNIX және MS – DOS файлдық жүйелерінде файлдың қарапайым логикалық құрылымы – бір байтты жазбалардың реттілігі болады.



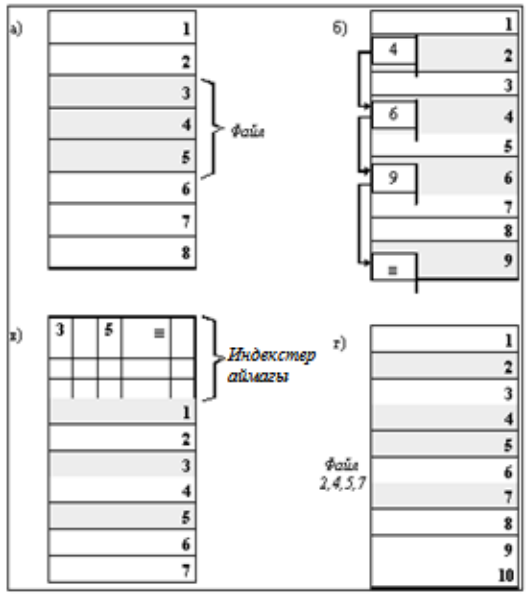
Сурет 6.2 - Файлдық жүйелердің логикалық ұйымдастырылуы.
а-бірдеңгейлі, б-иерархиялық (ағаш), в-иерархиялық (желі)

6.2.2. Физикалық ұйымдастыру және файл адресі

Файлдарды физикалық ұйымдатыру сыртқы жады құрылғысында, көпшілік жағдайларда дискіде файлдарды орналастыру ережесін сипаттайды. Файл физикалық жазбалардан – блоктардан тұрады. Блок – сыртқы құрылғы_жедел жадымен алмасатын мәліметтердің ең кіші бірлігі. Үздіксіз орналастыру – файлға дискілік жадының біртұтас тұтас бөлігін түзетін диск блоктарының реттілігін беретін ең қарапайым нұсқа (сурет 6.4, а), физикалық ұйымдастырудың ең қарапайым нұсқасы. Бұл жағдайда, файлға адрес беру үшін, тек қана бастапқы блоктың нөмірін көрсету жеткілікті. Бұл әдістің келесі артықшылығы - қарапайымдылығы. Бірақ, екі айтарлықтай кемшілігі бар. Біріншіден, файлды жасау уақытында оның ұзындығы ертерек белгісіз болады, сондықтан бұл файл үшін қанша жады тіркеу керек екендігі белгісіз болады. Екіншіден мұндай орналастыру тәртібінде міндетті түрде фрагменттеу пайда болады да, диск кеңістігі тиімді пайдаланылмайды, сөйтіп жеке кішірек өлшемді (ең аз дегенде 1 блок) бөліктері пайдаланылмай қалып қоюы мүмкін.

Сурет 6.3 - Файлдардың логикалық ұйымдастырылуының әдістері

Физикалық ұйымдастырудың келесі әдісі – дискілік жады блоктарының байланысқан тізімі түрінде орналастыру (сурет 6.4, б). Мұндай тәсілде әрбір блоктың басында келесі блокқа көрсететін нұсқағыш болады. Бұл жағдайда да файл адресі бір санмен – бірінші блоктың нөмірімен берілуі мүмкін болады. Алдыңғы тәсілден айырмашылығы ,әрбір блок қандай да бір файл тізбегіне байланыстырылады, сондықтан фрагменттеу болмайды. Файл өзінің қызмет етуі уақытында блок санын өсіре отырып, өзгеруі мүмкін. Кемшілігі файлдың еркін таңдалған орнына қатынауды іске асырудың күрделілігінде болып тұр: файлдың рет бойынша бесінші блогын оқу үшін, блок нөмірлерінің тізбегін сақтай отырып, бірінші төрт блокты ретімен оқып шығу қажет. Мұнан басқа, бұл әдісте бір блоктағы файл мәліметтерінің мөлшері екілік дәрежесіне (бір сөз келесі блоктың нөміріне жұмсалған) тең емес, бірақ көп бағдарламалар мәліметтерді, екілік дәрежесіне тең блоктармен оқиды.



Сурет 6.4 - Файлдарды физикалық ұйымдастыру
А) үздіксіз орналастыру Б) блоктардың байланысқан тізімі
В) индекстердің байланысқан тізімі Г) блок нөмірлерінің тізімі

MS – DOS операциялық жүйенің FAT файлдық жүйесінде индекстердің байланысқан тізімін пайдалану, кең таралған әдістің мысалы бола алады. Әрбір блокпен қандай да бір элемент – индекс байланыстырылады.

Индекстер дискінің жеке аймақтарында орналасады. (MS – DOS та бұл FAT кестелері). Егер қандай да бір блок қандай да бір файлда таратылған болса, онда бұл блоктың индексінде осы файлдың келесі блогының нөмірі болады. Мұдай физикалық ұйымдастырылуларда өткен әдістің барлық, артықшылықтары сақталады, бірақ белгіленген екі кемшілікте алынып тасталады: біріншіден, файлдың еркін таңдалған орнына қатынау үшін, тек қана блок индекстерін оқу, тізбек бойынша файл блоктарының керек мөлшерін санау және керек блоктың нөмірін анықтау жеткілікті, екіншіден файл мәліметтері блокты толығымен алады, яғни екілік дәрежесіне тең көлемге ие болады.

6.2.3. Файлға қатынау құқығы

Файлға қатынвау құқығын анықтау – әрбір қолданушы үшін, ол осы файлға қолданатын операциялар жиынын анықтау деген сөз. Әр түрлі файлдық жүйелерде өзінің дифференциалданған қатынау операцияларының тізімін анықтау мүмкін болады. Бұл тізімге келесі операциялар кіруі мүмкін:

- файл құру;
- файлды жою;
- файлды ашу;
- файлды жабу;
- файлды оқу;

- файлды жазу;
- файлды толықтыру;
- файлды толықтыру;
- файлды іздеу;
- файл атрибуттарын алу;
- атрибуттардың жаңа мәндерін орнату;
- орын ауыстыру;
- файлды орындау;
- каталогты оқу, және басқа операциялар.

Жалпы жағдайда қатынас құру құқығы, бағандары жүйенің барлық қолданушыларға сәйкес, ал жолдар мен бағандардың қиылысуында рұқсат етілген операциялар көрсетілетін, қатынас құру құқығы матрицасымен сипатталуы мүмкін болады. (сурет 6.5). Кейбір жүйелерді қолданушылар жеке санаттарға бөлінуі мүмкін. Бір санаттың барлық қолданушыларына біртұтас қатынау құру құқығы анықталады. Мысалы: UNIX жүйесінде барлық қолданушылар үш санатқа бөлінеді: файл иелері, ол топтың мүшелері және қалған барлықтары.

Қолданушы атауы	Файлдар атауы			
	modern.txt	win.exe	class.dbf	unix.ppt
kira	оқу	орындау	–	орындау
genya	оқу	орындау	–	оқу орындау
nataly	оқу	–	–	оқу орындау
victor	оқу жазу	–	құру	–

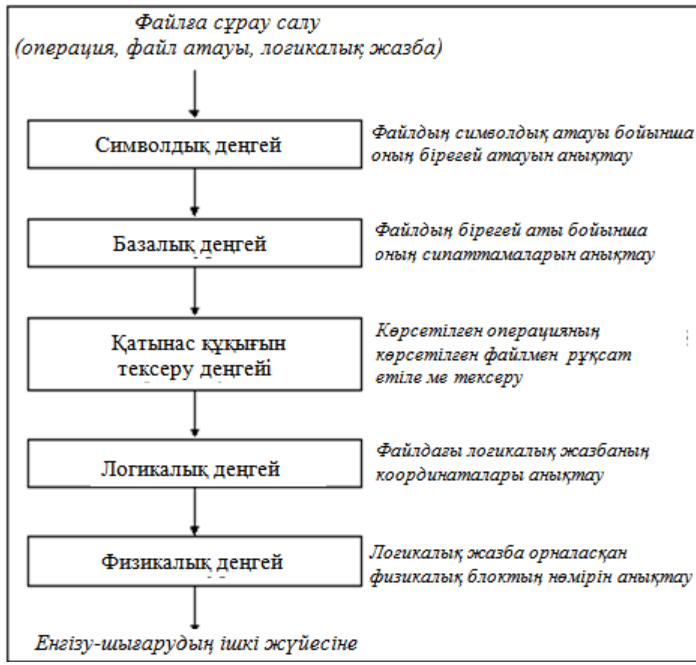
Сурет 6.5 – Қаынау құқығының матрицасы

Қатынау құру құқығын анықтауда екі негізгі әдісті бөледі:

- таңдау бойынша қатынау құру, мұнда әрбір файл және әрбір қолданушы үшін рұқсат етілген операцияларды иесінің өзі анықтайды;
- мандаттық қатынау, мұнда жүйе қолданушының қай топқа жатқызылғандығыпа байланысты, оның әрбір бөлінетін ресурстарға (бұл жағдайда файл) қатынасын белгілі бір құқықтармен қамтамасыз етеді.

6.2.4. Файлдық жүйенің жалпы моделі

Кез келген файлдық жүйенің қызмет етуін көп деңгейлі модельмен көрсетуге болады (сурет 6.6), онда әрбір деңгей өзінен жоғары жатқан деңгейге қандай да бір интерфейс (функциялар жиынтығын) береді, ал өзі, өз кезегінде , өзінің жұмысын орындау үшін өзінен төменірек жатқан деңгейдің ин.терфейсін пайдаланады (сұраныс жиынтығымен қатынас жасайды).



Сурет 6.6 – Файлдық жүйенің жалпы моделі

Символдық деңгейдің міндеті файлдың символдық атауы бойынша , оның бірегей атауын анықтау болып табылады. Әрбір файл тек бір ғана символдық атқа (мысалы: MS – DOS) ие болатын файлдық жүйелерде бұл деңгей болмайды, өйткені, қолданушының файлға берген аты бір мезгілде бірегей атау болып және операциялық жүйе пайдалануы мүмкін. Бір файлдың бірнеше символдық аты бола алатын басқа файлдық жүйелерде, мұндай деңгейде файлдың бірегей атын анықтау үшін каталогтар тізбегі қарастырылады.

Мысалы: UNIX файлдық жүйесінде файлдың индекстік дескриптор нөмірі (I – node) оның бірегей аты болып табылады.

Келесі базалық деңгейде файлдың бірегей аты бойынша , оның сипаттамалары анықталады: қатынас құру құқығы, адресі, өлшемі және басқалары. Бұрын айтылып кеткеніндей, файл сипаттамалары каталог құрамына кіруі немесе жеке кестелерде сақталуы мүмкін.

Файлды ашқанда, файлға түсудің орташа уақытын азайту үшін, оның сипаттамалары дискіден оперативті жадыға ауыстырылады. Кейбір файлдық жүйелерде (мысалы: HPFS) файлды ашқанда, оперативті жүйеге оның сипаттамаларымен бірге мәліметтер жинақталған файлдың бірнеше бірінші блоктары көшеді.

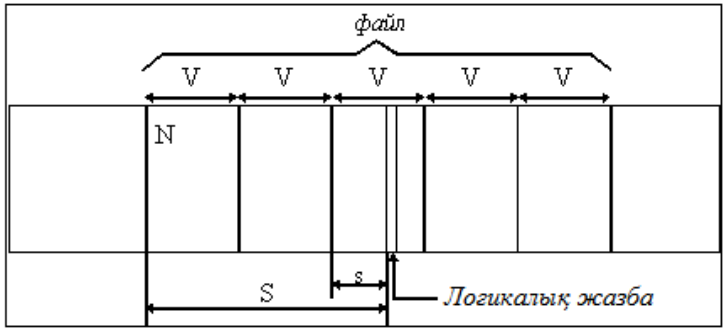
Файлға сұранысты іске асырудың келесі кезеңі оған қатынас құру құқығын тексеру болып табылады. Бұл үшін қолданушының немесе сұранысты берген үдерістің өкілеттіліктері берілген файлға қатынау құқығының рұқсат етілген түрлерінің тізімімен салыстырылады. Егер, үдерістің сұраныс салынған қатынас құру түрі рұқсат етілген болса,

онда сұранысты орындау жалғасады, егер рұқсат етілмеген болса, онда қатынас құру құқығы бұзылғандығы туралы хабарлама беріледі.

Логикалық деңгейде,сұраныс салынған файлдағы логикалық жазбаның координаталары анықталады, яғни талап етілген логикалық жазба файлдың басынан қандай қашықтықта (байтпен) екендігін анықтау талап етіледі. Мұнда файлдардың физикалық орналасуынан абстракцияланады ол байттардың үздіксіз реттілігі түрінде көрсетіледі. Берілген деңгейдің жұмыс алгоритмі файлдардың физикалық ұйымдастырылуына байланысты. Мысалы: егер файл 1 тиянақталған ұзындықтағы логикалық жазба реттілігі сияқты ұйымдастырылса, онда n – ші логикалық жазбаның ығысуы 1(n – 1) байт болады. Индексті – реттілікпен ұйымдастырылған файлды логикалық жазба координатасын анықтау үшін, логикалық жазбаның адресі тікелей көрсетілген, индекстер (кілттер) кестесін оқу орындалады.

Физикалық деңгейде файлдық жүйе талап етілген логикалық жазба орналасқан физикалық блоктағы нөмірін және физикалық блоктағы логикалық жазбаның ығысуын анықтайды. Бұл міндеттерді шешу үшін логикалық деңгей жұмысының нәтижесі – логикалық жазбаның файлдағы ығысуы,сыртқы құрылғылардағы файлдың адресі, сол сияқты файлдың физикалық ұйымдастырылуы туралы мәліметтер (блок өлшемі қоса кіреді) пайдаланылады.

Файлдың қарапайым физикалық ұйымдастырылуы үшін физикалық деңгейдің жұмысын блоктардың үздіксіз реттілігі түрінде көрсетіп, иллюстрациялайды. Физикалық деңгей міндеттері файлдың логикалық ұйымдастырылуы қалай болғандығына байланыссыз шешіледі.



Сурет 6.7 - Файлдық жүйенің физикалық деңгейінің функциялары

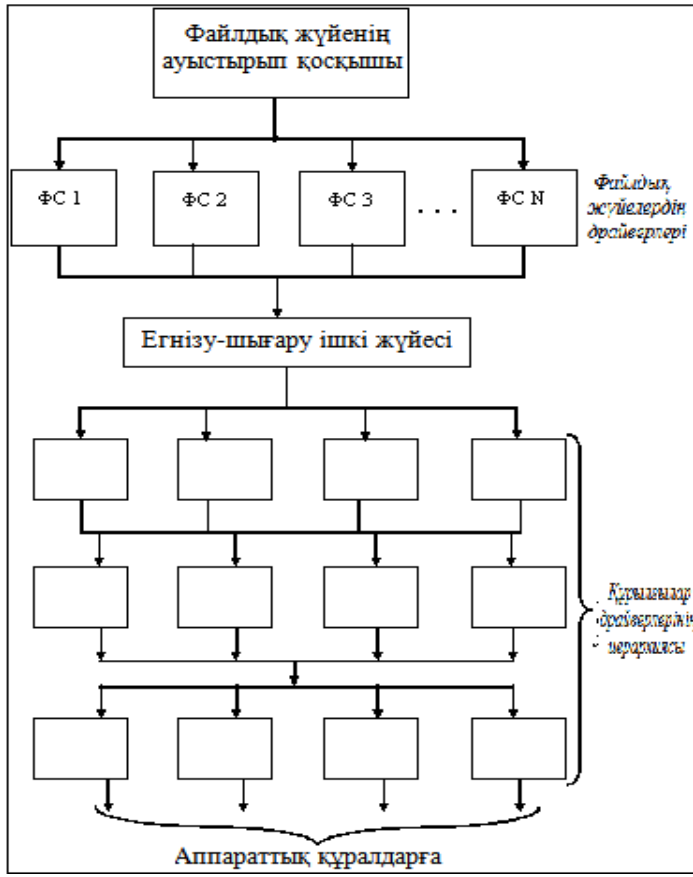
Файлдық жүйе физикалық блоктың нөмірін анықтағаннан кейін, сыртқы құрылғымен ауысу операциясын орындау үшін, енгізу - шығару жүйесіне қатынайды. Бұл сұраныс салуға жауап ретінде , физикалық деңгейдің жұмысы кезінде алынған ығысу негізінде, талап етілген логикалық жазба таңдалатын, керекті блок файлдық жүйенің буферіне берілетін болады.

6.2.5. Файлдық жүйелердің қазіргі архитектурасы**.**

Жаңа операциялық жүйе құрастырушылар қолданушыны бірден бірнеше файлдық жүйемен жұмыс істеу мүмкіндігімен қамтамасыз етуге ұмтылады.

Жаңаша түсінуде, файлдық жүйе оның құрымында әдеттегі түсініктегі файлдық жүйеде кіретін, көптеген құрауыштардан тұрады.

Жаңа файлдық жүйе, жоғарғы деңгейінде файлдық жүйенің ауыстырып – қосқыштары деп аталатындар орналасқан, көпдеңгейлі құрылымды болып келеді(Windows 95-те, мысалы, диспетчер) орнатқан файлдық жүйе - installable filesystem manager, IFS) осындай ауыстырып – қосқыш деп аталады. Ол сұраныс салу қосымшасы мен осы қосымша қатынас жасайтын нақты файлдық жүйенің арасында интерфейс қамтамасыз етеді. Файлдық жүйенің ауыстырып – қосқыштары келесі деңгейлер – файлдық жүйе деңгейлері қабылдайтын сұраныс салуларды форматқа түрлендіреді.



Сурет 6.8 - Заманауи файлдық жүйенің архитектурасы

Файлдық жүйе деңгейлерінің әрбір құрауышы сәйкес файлдық жүйесінің драйвері түрінде орындалады және файлдық жүйесінің белгілі бір ұйымдастырылуын сүйемелдейді. Ауыстырып – қосқыш файлдық жүйе драйверіне қатынас ете алатын жалғыз модуль. Қосымша оған тікелей қатынас ете алмайды. Файлдық жүйе драйвері реентерабельді код түрінде жазыла алады, бұл бірнеше қосымшаның бірден файлдармен операция орындауын

мүмкін етеді. Файлдық жүйенің әрбір драйвері өзінің инициализациялануында, файлдық жүйеге келесі қатынас жасағанда пайдаланатын кіру нүктелерінің кестесін ауыстырып қосқышқа бере отырып, онда тіркеледі.

Файлдық жүйе драйверлері өзінің функцияларын орындау үшін, файлдық жүйенің жаңа архитектурасының келесі қабатын түзетін, енгізу – шығару ішкі жүйесіне қатынас етеді. Енгізу -шығару ішкі жүйесі – бұл жүтеу инициализациялау және файлдық жүйенің төменгі деңгейінің барлық модульдарын басқару үшін жасап беретін файлдық жүйенің құрамды бөлігі. Әдетте, бұл модульдар тікелей ақпараттық құралдармен жұмыс істейтін порт драйверлері болып табылады. Мұнан басқа енгізу – шығару ішкі жүйесі, файлдық жүйе драйверлерін олардың нақты құрылғыға сұраныс салуын іске асыруын мүмкін ететін, қандай да бір сервиспен қамтамасыз етеді. Енгізу – шығару ішкі жүйесі жадыда тұрақты болуы және құрылғы драйверлерінің иерархиясының бірлескен жұмысын ұйымдастыруы керек. Бұл иерархияға белгілі бір түрдегі (тип) құрылғы драйверлері кіруі (қатты диск драйверлері немесе таспаға жинақтауыштар) жабдықтаушылар сүйемелдейтін драйверлер (мұндай драйверлер блоктық құрылғыларға келген сұраныс салуды қағып алады және бұл құрылғының қызмет көрсетіп тұрған драйверінің тәртібін аздап өзгертуі мүмкін, мысалы, мәліметтерді шифрлеуі) нақты адаптерлерді басқаратын порт драйверлері.

Файлдық жүйе архитектурасының деңгейлер санының көптігі, құрылғы драйвері авторларын үлкен иілмелілікпен қамтамасыз етеді – драйвер сұраныс салуда орындаудың кез келген кезеңінде басқару ала алады. Файлдармен жұмыс істеумен айналысатын қосымша функцияларының шақыруынан бастап, ең төменгі деңгейде жұмыс істейтін құрылғы драйвері контроллер регистрін қарап шығуды бастайтын уақытқа дейінгі уақыт**.** Файлдық жүйе жұмысының көпдеңгейлілік механизмі шақырулар тізбегінің көмегімен жүзеге асырылады.

Инициализациялау барысында құрылғы драйвері қатынас жасау келесі деңгейін анықтап алып, өзін қандай да бір құрылғының шақырулар тізбегіне қоса алды. Енгізу – шығару ішкі жүйесі мақсатты функцияның адресін, берілген деңгейді, тізбекті жеткілікті түрде реттеу үшін қолдана отырып, құрылғының шақыру тізбегіне орналастырылады. Сұраныс салуды орындау барысында енгізу – шығару шкі жүйесі, ертеректе шақырулар тізбегіне орналастырылған барлық функцияларды реттілікпен шақырады.

Шақырулар тізбегіне енгізілген драйвер процедурасы сұраныс салуды ары қарай деңгейге келесі деңгейге өзгертілген немесе өзгертілмеген түрде беруді шешеді, немесе, егер бұл мүмкін болса, процедура сұраныс салуды тізбек бойынша ары қарай бермей – ақ, оны қанағаттандырады.

6.3 Бақылау сұрақтары

6.3.1 Файлдық жүйе дегеніміз не?
6.3. 2 Файлдың жүйенің функциялары қандай?
6.3.3 Файлдардың қандай түрлері болады?
6.3.4 Файл атрибуттары дегеніміз не?
6.3.5 Каталогтардың құрылымы қандай?
6.3.6 Файлдардың логикалық және физикалық ұйымдастырылуын сипаттаңыздар.
6.3.7 Файлдарға қатынас құрудың қандай құқықтары болады?
6.3.8 Файлдық жүйенің жалпы моделін сипаттаңыздар.
6.3.9 Файлдық жүйелердің архитектурасы қандай?

7 WINDOWS XP/2000 АРХИТЕКТУРАСЫ

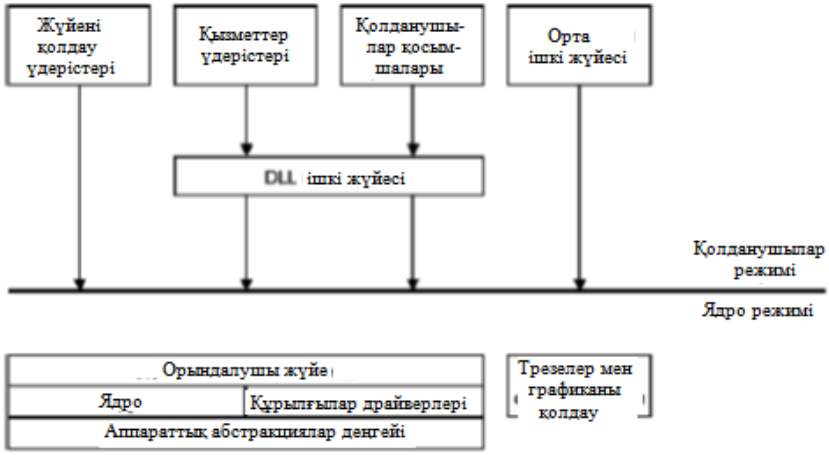
7.1 Дәріс мақсаты

Дәріс мақсаты Windows XP/2000 архитектурасымен танысу болып табылады.

7.2 Теориялық мәліметтер

Windows 2000 ОЖ енгізу-шығару жүйесі ОЖ орындаушы жүйе компоненттерінен тұрады. Олар аппараттық құралдарды басқарады және жүйеге оларға қатынауға, сонымен қаар қолданушылар бағдарламаларына (приложения) қатынауға интерфейстер ұсынады. Қолданбалы бағдарламалық қамтамасыздандырудың аппараттық қамтамасыздандырумен қатынасуының басты құралы драйверлер болып табылады. Windows 2000 ОЖ түрлі драйверлер қолданылады, олардың ОЖ, жүйелік компонентер және қолданушылардың бағдарламаларымен әрекеттесуі әр түрлі болып келеді.

Windows 2000 ОЖ қысқартылған архитектурасы сурет 7.1 берілген.



Сурет 7.1 - Windows 2000 қысқартылған архитектурасы

ОЖ элементтері екі класқа бөлінеді: Біреулері қолданушы режимінде, енді бірлері ядро режимінде орындалады.

Windows 2000 ОЖ қолданушы үдерістері үдерістердің қорғалған адрестік кеңістігінде орындалады (яғни, ядро режимінде орындаған ақытта олар жүйелік кеңістікке қолжетімділік алады).

Windows 2000 ОЖ қолданушы үдерістерінің 4 түрі бар:

- Жүйені қолдайтын тиянақталған үдерістер (System Support Processes), мысалы, Windows 2000 ОЖ қызметтері болып табылмайтын, жүйеге енуді өңдеу үдерісі және сеанстар диспетчері;
- Қызметтердің үдерістері (Service processes) - Win32-сервистерін тасымалдаушылар, Task Scheduler (тапсырмаларды жоспарлаушы) және Spooler (баспа спулері) сияқтылар;

- қолданушы бағдарламалары, (User Applications) – бес типті болады: Win32, Windows 3.1, MS-DOS, POSIX и OS/2 1.2;

- ортаның ішкі жүйесі (Environment Subsystems) – қолданушы бағдарламалрына ОЖ ендірілген қызметтерді шақырылатын функциялар арқылы ұсынады, осылай олар ОЖ ортасын құрады.

«DLL ішкі жүйесі» элементіне назар аудару керек. Бұл элементтің болуы Windows 2000 ОЖ қолданушы бағдарламалары ОЖ енірілген қызметтерін (жүйелік қызметтерді) тікелей шақыра алмайды, олар бір немесе бірнеше DLL ішкі жүйесі (Subsystem DLL) арқылы жұмыс істейді. Олар құжатталған функцияларды сәйкес Windows 2000 ОЖ жүйелік қызметтерінің құжатталмаған ішкі шақыруларына трансляциялауға арналған.

Windows 2000 ОЖ ядро режмінде келесі компоненттері бар:

- орындалатын жүйе (executive) – ОЖ базалық қызметтері бар (жадыны, үдерістерді және ағындарды басқаруды, енгізу-шығаруды және үдерістер арасындағы өзара қатынасты қамтамасыз етеді);

- ядро (kernel) - жоғарғы деңгейдің құрылымдарын жүзеге асыру үшін орындалатын жүйе қолданатын процедуралар мен негізгі объектілердің жиынын ұсынады; ОЖ төменгі деңгейлі функциялары бар (ағындарды жобалау, үзілістер, үдерістерді синхронизациялау және т.б. қолдау);

- құрылғылар драйверлері (Device Drivers) - қолданушылардың шақырулары енгізу-шығару функцияларын нақты бір құрылғыға арналған арнайы сұратуларға трансляциялайды;

- аппараттық абстракция деңгейі (Hardware Abstraction Layer, HAL) - ядро, драйверлер және Windows 2000 орындалатын жүйесін сыртқы құрылғылармен аппараттық платформадан оқшаулайды;

- терезелерді және графиканы қолдаудың ішкі жүйесі (Windowing and Graphics System) - графикалық қолданушы интерфейсін (GUI) жүзеге асырады.

Кесте 7.1 –Windows 2000 жүйелік компоненттерінің негізгі файлдары

Файл атауы	Жүйе компоненті
Ntoskrnl.exe	орындаушы жүйе және ядро
Hal.dll	аппараттық абстракция деңгейі
Win32k.sys	Ядро режимінде жұмыс жасайтын, Win32 ішкі жүйесінің бөлімі
Ntdll.dll	Ішкі қолдау функциялары және орындалушы функциялары бар жүйелік қызметтер диспетчерінің интерфейчтері
Kernel32.dll, Advapi32.dll, User32.dll, Gdi32.dll	Win32-нің негізгі DLL ішкі жүйелері

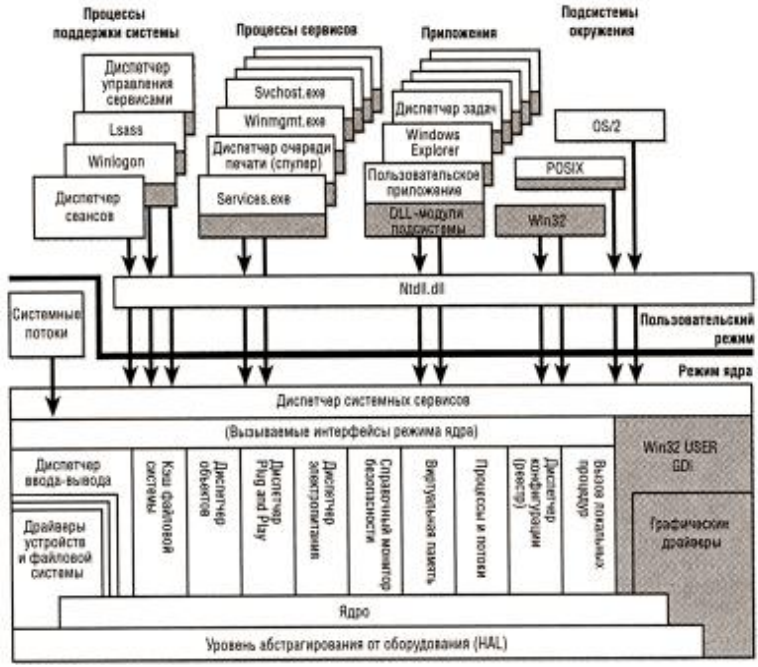
Windows 2000 ОЖ үш ортаның ішкі жүйесі бар: Win32, POSIX және OS/2. Windows 2000 ОЖ Win32 ішкі жүйесінсіз жұмыс істей алмайды. Бұл ішкі жүйе пернетақта, тінтуір (mouse), экранмен байланысты операциялардың барлығын өңдейді. Бұл ішкі жүйе тіпті интерактивті қолданушылары жоқ серверлерге де қажет. Win32 ішкі жүйесі қашанда жұмыс істеп тұрады, ал басқа ішкі жүйелер тек қажет болғанда ғана іске қосылады.

Windows 2000 ОЖ іске қосылғанда іске қосқан уақытта қандай ішкі жүйелер жүктелетіндігі Required параметрінің көмегімен, реестрдің бөлімінде

HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Subsystem анықталады. Бұл параметрде тек іске қосылғанда жүктелінетін ішкі жүйелер тізімі анықталады.

Ортаның әрбір ішкі жүйесі қолданбалы бағдарламаларға Windows 2000-тің орындалатын жүйесінің базалық қызметтері ішінен өз жиынтығын ұсынады. Яғни, Windows 2000 операциялық жүйесінде бір ішкі жүйе үшін құрылған бағдарлама тек осы ішкі жүйенің ғана операцияларын орындай алады және басқа ішкі жүйенің операцияларын орындай алмайды. әрбір орындалатын файл (.EXE) тек бір ішкі жүйеге ғана тиесілі бола алады.

Қолданушы қолданушы бағдарламалары жүйелік қызметтерді тікелей шақыра алмайды, олар тек DLL ішкі жүйелерге ғана қатынай алады. Бұл DLL бағдарламамен ол шақырған ішкі жүйе арасында құжатталған интерфейті (документированный интерфейс) ұсынады. Win32 (Kernel32.dll, Advapi32.dll, User32.dll, Gdi32.dll) DLL ішкі жүйесі Win32 API функцияларын жүзеге асырады, ал POSIX DLL ішкі жүйесі POSIX 1003.1 API функциясын жүзеге асырады.



Сурет 7.2 –Windows 2000 архитектурасы

7.2.1 Ортаның ішкі жүйелері

Windows 2000 ОЖ қрамына үш ортаның ішкі жүйесі кіреді: Win32, POSIX және OS/2.

Win32 ішкі жүйесі келесі негізгі элеменеттерден тұрады:

- ортаның ішкі жүйесінің үдерісі , консольдық (мәтіндік) терезелерді, үдерістермен ағындарды құру мен жоюды қолдау, GetTempFile, DefineDosDevice, TxitWindowsEx типті функциялар;

- ядро режимінің драйвері құрамына (Win32k.sys), терезелерді экранда қайта салу және шығару, пернетақта, тінтуір және т.б. құрылғылардан енгізуді қабылдау, сонымен қатар қолданушылардың хабарламаларын бағдарламаларға жіберуді басқаратын терезе диспетчері кіреді;
- ішкі жүйелердің DLL-модульдері (Kernel32.dll, Advapi32.dll, User32.dll, Gdi32.dll), Win32 API фунцияларының құжатталған шақыруларын сәйкес Ntoskrnl.exe және Win32.sys-ден ядро режимінің құжатталмаған қызметіне трансляциялайды;
- графикалық құралдар драйверлері – нақты бір құрылғыларға арналған дисплей, принтер драйверлері және видеокарта минипорт-драйверлері түрінде болады.

POSIX ішкі жүйесі (Portable Operating System Interface Based on UNIX –UNIX негізіндегі ОЖ жылжытылатын интерфейсі) – бұл UNIX типті ОЖ интерфейстерінің халықаралық стандарттарының жиынтығы. Бұл POSIX бағдарламаларына бастапқыдан қолжетімді функциялар жиынтығы, POSIX.1 стандартымен анықталған қызметтермен шектелген. Бұл шектеулердің мәні, POSIX бағдарламасы ағын немесе Windows 2000 терезесін құра алмайды, сонымен қатар RPC (Remote Procedure Call (желілік бағдарламалау стандарты, ол бірнеше процедуралардан тұратын, олардың біразы – локальді, ал тағы біразы - басқа (қашықтағы) компьютерлерде орындалатын бағдарламалар құруға және сокеттер (коммуникаациялық байланыстардың ақырғы нүктесі) мүмкіндік береді)) қолдана алмайды.

POSIX ішкі жүйесінің OS/2 ішкі жүйесі сияқты функционалдық қызметтері шектеулі.

7.2.2 Орындалатын жүйе

Орындалатын жүйесі (Executive) Ntoskrnl.exe (ядро төменгі деңгейде орналасқан) жоғарғы деңгейінде орналасқан. Оның құрамына келесі функциялар кіреді:

– экспортталатын функциялар, қолданушы режимінде шақыруға қолжетімді және жүйелік қызметтер деп аталады; қызметтердің көбі Win32 API немесе API ортаның ішкі жүйелері арқылы қолжетімді;
– ауқымды ретінде анықталған функциялар, бірақ экспортталмайды;
– қандай да бір модульдегі ішкі функциялар, бірақ ауқымды болып анықталмаған.

Орындалатын жүйе келесі негізгі компоненттерден тұрады:

- конфигурация диспетчері, жүйелік реестрді басқарады;
- үдерістермен ағындар диспетчері, үдерістермен ағындарды құрып және оларды аяқтайды;
- қауіпсіздіктің анықтамалық мониторы, локальді компьютерде қауіпсіздік саясатын жүзеге асырады (ОЖ ресурстарын күзетеді және орындалу барысында объектілерді бақылайды);
- енгізу-шығару диспетчері, аппаратты-тәуелсіз енгізу-шығаруды жүзеге асырып, енгізу-шығаруды ары қарай өңдеу үшін құрылғылардың керекті драйверлеріне бағыттайды;
- Plug and Play диспетчері, нақты бір құрылғыны құптау үшін қажетті драйверлерді анықтау және оларды жүктейді; әрбір құрылғының аппараттық ресурстарға қоятын талаптары құрылғыларды тізбелеу барысында анықталады;
- Электрмен қуаттандыру диспетчері, электрмен қуаттандырумен байланысты оқиғаларды реттеп, драйверлер үшін электрді реттеу жүйесінің ескертулерін генерациялайды;
- WMI ішкі бағдарламасы (Windows Management Instrumentation –Windows басқару инчтрументарии) драйверлерге өзінің жұмыс сипаттамалары және конфигурация туралы ақпаратты жариялауға мүмкіндік береді, сонымен қатар қолданушы режимінде WMI қызметінен командалар алады;
- Виртуалды жады диспетчері, виртуалды жадыны жүзеге асырады – әрбір үдеріске көлемі қолжетімді физикалық жады көлемінен үлкен болуы мүмкін жабық адрестік кеңістік бөлуге мүмкіндік беретін жадыны басқару сұлбасы.

7.3 Бақылау сұрақтар

7.3.1 XP/2000 архитектурасы қандай?
7.3.2 Windows XP/2000 қолданушы үдерістерінің қандай типтері бар?
7.3.3 Windows XP/2000 ядро режимінің қандай компонеттері бар?
7.3.4 Windows XP/2000 қандай орта ішкі жүйелері бар?
7.3.5 Ntdll.dll модулі не үшін арналған?
7.3.6 Орындалатын жүйенің құрамына қандай функциялар кіреді?
7.3.7 Орындалатын жүйе қандай компоненттерден тұрады?
7.3.8 Ядро неге арналған?
7.3.9 Аппараттық абстракция деңгейі не үшін арналған?
7.3.10 Енгізу/шығару ішкі жүйесі қандай компоненттерден тұрады?
7.3.11 Windows XP/2000 құрылғылар драйверінің қандай түрлері бар?

8 LINUX ОПЕРАЦИЯЛЫҚ ЖҮЙЕСІ

8.1 Дәріс мақсаты

Дәріс мақсаты LINUX операциялық жүйесі архитектурасымен таныстыру болып табылады.

8.2 Теориялық мәліметтер

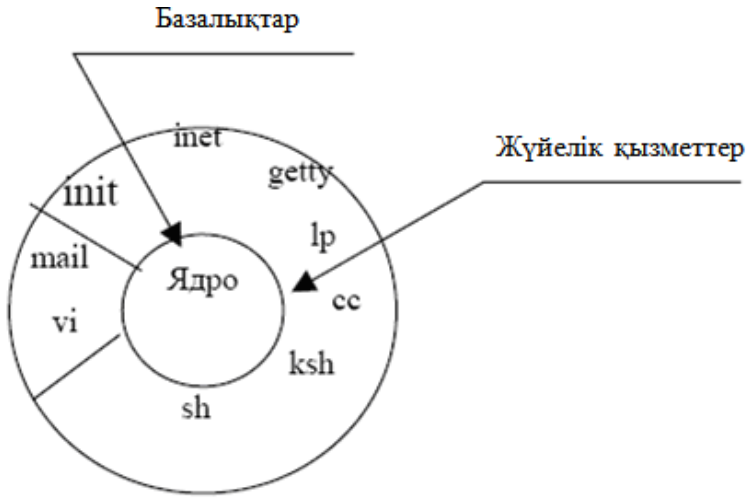
1 қаңтар 1970 жыл ресми түрде UNIX операциялық жүйесінің туған күні болып саналады. Тура осы мезеттен бастап кез келген UNIX жүйесі өзінің жүйелік уақытын есептеуді бастайды. Бұл – операциялық жүйе үшін өте үлкен мерзім. Бүгін, әкімшілік ету көзқарасы тұрғысынан алғанда, неғұрлым қарапайым және ыңғайлы жүйелердің пайда болуына қарамастан, UNIX барлық көшбасшылардың арасында мықты орын алады.

UNIX- тің негізгі сипаттамаларына келесілерді жатқызуға болады:

- Жүйе коды СИ тілінің жоғары деңгейінде жазылған,сондықтан бұл оны түсінуді, өзгертуді және басқа аппарат платформаларына көшіруді қарапайым етеді:
- UNIX – көпміндетті, көпқолданушылы жүйе. Бір мықты сервер қолданушылардың көп мөлшерінің сұранысына қызмет көрсете алады. Жүйе ең көп әр түрлі функцияларды орындай алады: жүздеген қолданушыларға қызмет көрсететін есептеуіш сервер ретінде, мәліметтер қорының сервері сияқты, желілік сервер сияқты немесе желілік бағыт көрсетуші маршрутизатор сияқты қызмет етеді
- UNIX нұсқасының көптүрлілігіне қарамастан, біркелкі архитектура және бірқатар стандарт қалыпталған интерфейстер олардың жанұясының негізі болып табылады.
- UNIX қарапайым, бірақ та, өте қуатты стандарт қалыпталған қолданушылықты интерфейстерге ие.
- UNIX файлдық жүйесінің үйлестірілген интерфейсі тек қана, дискіде сақталған мәліметтерге қатынас құруды ғана емес және тағы да терминалға, принтерлерге, магниттік таспаларға, желілерге және тіпті, жадыға да қатынас құруды жүзеге асырады.
- UNIX жүйесі үшін, қарапайым мәтіндік редактордан бастап, мәліметтер қорын басқаратын қуатты жүйелерге дейінгі, әртүрлі қосымшалардың көптеген саны жасап шығарылған.

Жалпы түрде UNIX ОЖ екі деңгейлі модельмен, суретте берілгендей бейнеленуі мүмкін (сурет 8.1).

Ортасында жүйе ядросы (kernel) орналасады. Ядро, қолданбалы бағдарламаларды оның архитектурасының ерекшеліктерінен оқшаулай отырып, тікелей компьютердің аппараттық бөлігімен өзара әрекет етеді. Ядроның өзіне қолданбалы бағдарламаларға берілетін , қызмет көрсету жиынтығын жүзеге асыратын бағдарламалар енеді. Ядроның қызмет көрсетуіне, енгізу – шығару операциялары, үдерістерді жасау және басқару, үдерістерді синхронизациялау жатады. Модельдің келесі деңгейі UNIX ОЖ қолданушылықты интерфейсін қамтамасыз ететін жүйелік қызметтер. Ядромен де, қосымшалармен де, жүйелік міндеттермен де өзара әрекет сызбасы біркелкі болады.



Сурет 8.1 - UNIX жүйесінің жалпы моделі

Linux (лин – нукс деп айтылады) – Intel 80368 (және одан да жоғары) процесорлы компьютерде жұмыс істеу үшін (спроектированная) құрастырылған 32 – разрядты операциялық жүйе. Техникалық көзқарас тұрғысынан, Linux ONIX тің баламасы болып табылады, яғни бұл жүйе UNIX стандарттық командаларын өңдейді және UNIX бағдарламаларын іске қосады. Linux BSD және System V кейбір кеңейтілімдерімен алғанда POSIX спецификациясына сәйкес келеді.

Басында Linux жобасымен барлығы бір ғана адам айналысты, Линус Торвальдс (Linus Torbalds) ,сол уақытта ол Хельсинкиде Финляндия университетінің студенті болды. Ол өзінің, UNIX нұсқаларының біреуі үшін, атап айтқанда, PC үшін жасап шығарылған, UNIX- тің коммерциялық баламасы болатын, Minix операциялық жүйесінің альтернативі болуын қалады. Ол Linux- ті жүйе Minix –ке ұқсас болатындай етіп (Linux Minix – тің файлдық жүйесін қолдайды), бірақ одан да орнықтырақ жұмыс істейтін және тегін болатындай етіп жобалады. Көп уақыт бойы Linux жасап шығару сатысында орналасқан операциялық жүйе болды. 1991 жылдың ортасында 0.2 нұсқасы шығарылды. Ары қарай Linux – пен жұмыс істеуде Торвальдске энтузиастар - еріктілер көмектесті. Мұнан басқа, бұл энтузиастар («Linuxті орнату және кофигурациялау» кітабының авторларының бірі – Патрика Фолькердингті қоса алғанда) біраз дәрежеде Linux танымалдығының артуына әсер ететін қосымша бағдарламалар құруға көмектесті. Linux (UNIX ке де қатысты) фундаментальды принципі мынада: Linux құралдар жиынтығы болатындығында.

Бір құрал бір міндеттер үшін, басқасы – басқа міндеттер үшін қолданылады.

Linux Интернет және желілер үшін тұрғызылған. Linux толығымен ашық жүйе болып табылады. Ұсынылған CD ROM да Slackware Linux операциялық жүйесінің бастапқы мәтіндері орналасады. Егер сізге операциялық жүйеге өзіңіздің өзгертулеріңізді енгізу қажеттілігі туса, сіз мұны істей аласыз. Егер сізге шалғай құрылғыға драйвер қажет болса, сіз оны өзіңіз жаза аласыз.

Linux көпміндетті операциялық жүйе болып табылады. Linux бірмезгілде бірнеше міндетті орындай алады және әр түрлі міндетке әртүрлі артықшылық берілуі мүмкін, ығыстырылатын көпміндеттілікті жүзеге асырады.

Linux көпқолданушылықты операциялық жүйе болып табылады. Linux-ті серверге орнатуға болады және бір компьютерге бірнеше қолдаланушыны қосуға болады. Бұдан басқа, сізде бұрыннан Slackware Linux бар болғандықтан, бірнеше пайдаланушыға қызмет көрсету үшін, сіз қосымша рұқсат алуға төлемейсіз.

8.2.1 Операциялық жүйе функциялары.

UNIX операциялық жүйесі келесі функцияларды орындай отырып, компьютердің аппаратымен және бағдарламалық ресурстармен өзара әрекет етеді:

- жабдықтарды басқару;
- ресурстарды басқару;
- пайдаланушы интерфейстерін қолдау;
- ақпаратты енгізу және шығаруды орындау;
- жүйе мониторингі;
- копьютерлік желіде алыстан қатынас құруды;

қамтамасыз ету.

Жабдықтарды басқару

Қолданбалы бағдарламаларда компьютер жабдықтарын тікелей басқару мүмкіндігі жоқ. Жабдықтарды басқару функциясын тек қана операциялық жүйе, шалғай құрылғылардағы қосымшаларға және басқа бағдарламаларға қатынас құру мүмкіндігін көрсете отырып, (мұнда осы сияқты құрылғыларды басқару спецификациясын білу қажеттілігі жойылады) орындайды. Операциялық жүйеге салынған бағдарламаның нақты аппараттық іске асыруға тәуелсіздігінің концепциясы ,UNIX операциясының мобильділігін қамтамасыз ететін, маңызды элементтердің бірі болып табылады, бұл оны әртүрлі аппараттық конфигурацияларда кеңінен қолдануға мүмкіндік береді.

Ресурстарды басқару

UNIX операциялық жүйесі көпміндетті және көпқолданушылықты жұмыс режимін қолдай отырып, әртүрлі объектілермен жиірек жұмыс істейтін көптеген қолданушылардың арасында компьютер ресурстарын үлестіруді басқарады. Көпміндетті режим бір қолданушыға бір мезгілде бірнеше бағдарламамен жұмыс істеуге мүмкіндік жасайды, бұл жағдайда ОП және оперативті жады көптеген үдерістердің арасында бөлінеді. Көп қолданушылықты режим бір мезгілде бірнеше қолданушының арасында компьютер ресурстарын бөле отырып, олардың жұмыс істеу мүмкіндігін қамтиды. Мұның қатарында принтер немесе плоттер сияқты қымбат шалғай құрылғылар да бар. Мұндай жұмыс режимі әрбіреуінде бір қолданушы жұмыс істейтін, бірнеше машинаны біріктіріп жұмыс істегеннен гөрі, экономикалық жағынан тиімдірек көрінеді.

Қолданушы интерфейсін қолдау

Қолданушы интерфейсі – бұл қолданушылардың есептеуіш жүйемен интерактивті (диалог қамтамасыз ету) өзара әрекетін қамтамасыз ететін құрылғы. UNIK операциялық жүйесінің қазіргі заманғы нұсқаларылары интерфейстің бірнеше түрін қолдайды: командалық тіркес, мәзірлер және қолданушылықты графикалық интерфейс.

Командалық жол, әдетте жүйенің функцияларымен және командаларымен таныс қолданушы үшін ыңғайлы. Қолданушы интерфейстің мұндай түрімен жұмыс істегенде, әрбір команданы «көмектесуге» (бастапқыдан – бұл доллар белгісі) пернетақтадан енгізеді. Мұндай интерфейспен қамтамасыз ететін бағдарламалар қапшық деп аталады. Бұл интерфейс жүйенің кез келген командасын орындауға мүмкіндік береді.

Әдетте UNIX операциялық жүйесінің құрамына мыналар сияқты үш жүйе кіреді: Bourne shell (sh), korn shell (ksh) және Cshell (csh).

Мәзірлерді әдетте, операциялық жүйеде кейбір орнатулар жүргізу қажеттілігі туған кезде, жаңа қолданушылар немесе қолданушылар пайдаланады. Көбінесе, мұндай

интерфейс гид функциясын жүзеге асырады: ол жүйе орындайтын қандай да бір функцияны, қолданушының тізімінен (мәзір) таңдап алуын мүмкін етеді. Әдетте, мәзір пайдаланушыға бірнеше таңдау мүмкіндігін ұсына отырып, көпдеңгейлі мәтін түрінде ұйымдастырылады. Жүйелік әкімгерге конфигурациялау бойынша жұмысты орындауға және жүйені баптауға көмектесетін, sysadm бағдарламасы осындай интерфейстің мысалы бола алады.

Графикалық қолданушылықты интерфейс жаңаларға қалай бағдарланса, білікті қолданушыларға да солай бағдарланған. Ол компьютермен өзара әрекеттің бірнеше жолын қамтамасыз етеді: тиктограммамен бейленетін жүйе объектілерін шолу, дисплей экранындағы графикалық бейнені (пиктограмма) таңдау tracking ball құрылғысы немесе тінтуір көмегімен таңдау арқылы команданы орындау, UNIX ОЖ X.desktop жұмыс істеу үшін графикалық қолданушылықты интерфейс сияқты, арнайы X – терминалдарымен немесе X.Windows жүйесін, әдеттегі графикалық терминалдармен жұмыс істеу үшін жабдықтайды.

Енгізу және шығаруды орындау

ОЖ мәліметтерді енгізуді жиірек талап ететін бағдарламаларды жүктейді және орындайды, сонымен бірге өздері мәліметтерді шығаруды жүзеге асырады. Мәліметтерді енгізу пернетақтадан, тінтуір көмегімен жүзеге асырылуы мүмкін болады немесе модем арқылы түсіп сәйкесінше, терминал экранында, принтерде шығуы немесе модемге бағытталуы мүмкін. ОЖ қолданушы енгізген мәліметтерді, оларды бағдарлама қабылдайтындай түрге түрлендіріледі, ал шығатын мәліметтер қолданушыға түсінікті болатындай түрге келтіріліп түрлендіріледі.

Жүйе мониторингі

Ресурсты жұмыс барысында есептеуіш жүйелердің ресурсы тұрақты пайдаланылады, босатылады, қайтадан пайдаланылады, сондықтан олар тұрақты белсенді күйде болуы керек және бұл ресурстарды талап ететін үдерістер үшін қол жетімді болуы керек, ОЖ осы белсенділікті қадағалауы, ,қақтығыстарды шешуі тиіс және осы ресурстар босатылғаннан кейін, қайтадан үдерістерге қол жетімді болуы үшін кепілдік етуі тиіс. Бұл үшін UNIX ОЖ келесілерді орындайды: файлдық жүйеге қатынағанда мәліметтердің тұтастығын тексереді, үдерістерді басқарады және олардың жүйе ресурстарын пайдалануларын бақылайды, қателерді алдын-ала анықтауды диагностикалауды қамтамасыз етеді, дұрыс жұмыс істемей тұрған үдерістерді аяқтайды және жүйені тоқтатады. Жүйеде статистикалық ақпараттарды жинай отырып, жекелей ресурстарды жүктеуді мерзімді түрде бақылай алатын, утилит жиынтығы болады. Алынған мәліметтерді талдау, жүйелік әкімгерге жүйе жұмысындағы «осал орындарды» анықтауға және жоюға көмектесе алады.

Компьютерлік желіде қашықтан қатынау құруды қамтамасыз ету

UNIX ОЖ қолданушылардың есептеуіш желіде жұмыс істейтін басқа компьютерлердің ресурстарына қатынас құруын қамтамасыз етеді. Операциялық жүйенің құрамында қашықтағы компьютермен байланыс орнатуға қащықтағы жүйеге тіркелуге, желі компьютерлерінің арасында мәліметтерді беруді жүзеге асыруға, электронды поштаны пайдалануға мүмкіндік беретін желілік қосымшалар жиынтығы бар. UNIX, қашықтағы компьютерледің файлдақ жүйесіне қатынас құру үшін, операциялық жүйенің командаларын пайдалануға мүмкіндік беретін NFS(Network File System) желілік файлдық жүйені қолдайды.

8.2.2 UNIX операциялық жүйенің құрауыштары

Операциялық жүйе – бұл жүйедегі үдерістерді басқару функцияларын қамтамасыз ететін және жүйе аппаратураларын қолданушылардың арасындағы интерфейсті жүзеге асыратын, бағдарламалар жиынтығы.

Командалар жүйесі

UNIX ОЖ құрамында, ұйымдастыру міндеттерін орындайтын және мәліметтерді өңдейтін, қолданушы қоршауын нығайтатын бірнеше жүздеген командалар бар. Командалардың өздері мәліметтердің минималды енгізілуін талап ете отырып, белгілі бір функцияларды орындайтын және салыстырмалы жылдам орындайтын, кішірек бағдарламалар болып табылады. Көбірек бөлігі барлық қолданушыларға қолжетімді, бірақ та 100дің манындағы командалар тек қана, артықшылық берілген қоданушы болып табылатын, жүйе әкімгеріне ғана қолжетімді. Атқару барысында ақпаратты интерактивті енгізу мүмкіндігі бар кейбір командаларды кейде утилит деп атйды. Мәтін редакторы VI және электрондық поштаны басқару командасы mail утилиттің мысалы бола алады.

Қабықшалар

Әдетте, қабықшалар деп операциялық жүйе ядросымен қолданушылы интерфейсін қамтамасыз етуші интерактивті бағдарламаны айтады. Қабықша жүйеде тіркелген мезеттен бастап, ол жүйеден шыққан мезетке дейін қолданушының белсенді үдерісі болады. Бұл бағдарламалар команданық интерпретатор (кейде оларды командалық процессор деп атайды) болып табылады.

Әдетте UNIX ОЖ құрамында бірнеше қабықша қолданылады:

- Bourne shell (sh): негізгі стандартты қабықша UNIX;
- korn shell (ksh): кеңейтілген нұсқа, Bourne shell;
- C shell (csh): UNIX танымал қабықша, Беркли университеті жасап шығарған (BSD UNIX) UNIX танымал қабықша;
- Шектеулері бар қабықша (restricted shell – rsh және rksh);
- Жүйеге қатынас құруды шектеу қажет болатын пайдаланушылар үшін жасап шығарылған Bourne shell және korn shell (ksh) ішкі жиыны.

Ядро

Жүйе ядросы, базалық функцияларды қамтамасыз ететін опреациялық жүйенің орталығы болып табылады: үдерістерді жасайды және оларды басқарады, жадыны үлестіреді, файлдарға және шалғай құрылғыларға қатынас құруды қамтамасыз етеді.

Қолданбалы міндеттердің ядромен өзара әрекеті жүйелік шақырулардың стандартты интерфейсі көмегімен өтеді. Жүйелік шақырулардың интерфейсі базалық қызметтерге сұраныстар форматын анықтайды.

Үдеріс, ядро процедурасымен анықталатын, жүйелік шақырулардың көмегімен, ядроның базалық функциясын сұрайды. Ядро сұранысты орындайды және үдеріске қажетті мәліметтерді қайтарады.

Ядро үш негізгі ішкі жүйеден тұрады:

- үдерістерді және жадыны басқарудың ішкі жүйесі;
- файлдық ішекі жүйе;
- енгізу – шығару ішкі жүйесі.

Үдерістерді басқару ішкі жүйесінің модульі келесі фнкцияларды орындайды.

- Үдерістерді жасау және жою;
- Үдерістердің арасында жүйе ресурстарын үлестіру;
- Үдерістерді синхронизациялау;
- Үдеріс аралық өзара әрекет.

Ядроның үдерістерді жоспарлаушылар орындайтын арнайы функциясы, үдерістердің арасындағы жүйе ресурстары үшін күрестегі қақтығысты шешеді.

8.2.3 Қолданушы идентификациясы

Қолданушыны, жүйелік әкімгер жүйеге тіркегенде, тіркелетін атаумен идентификациялаудың екі құрауышы байланыстырылады: пайдаланушы идентификаторы (user ID – UID) және ол кіретін топ идентификаторы (group ID -GID).

Қолданушы атауы бірегей санмен байланыстырылады. Жүйе оны UNIX ОЖ әртүрлі қорғаныс механизміндегі құрал сияқты пайдаланда,мысалы,файлдарды қорғауда немесе артықшылық берілген командаларды орындағанда.

Кез келген UNIX ОЖ UID=0 мен байланысты, супер қолданушығы тиісті бір root арнайы атауы бар.

Бұл қолданушыда барлық жүйелік артықшылықтар бар дегенді білдіреді.

Сол сияқты топ атауы, ортақ міндеттермен біріккен, қолданушылар тобына қатысты санмен байланысты, мысалы: бөлім қызметкерлері, бір лек студенттері және тағы да басқалары. Сол сияқты, бұл санды жүйенің қорғаныс механизмдері пайдаланылады. Егер қолданушыға басқа топтардың мәліметтерімен жұмыс істеу қажет болса, бұл идентификатор басқа топтардың атуларымен байланыстырылады.

Жүйенің қолданушылары туралы барлық тіркеу ақпараттары /etc/ paswb файлында сақталды.

UNIX ОЖ қазіргі заманғы нұсқасында шифрленген парольдар және оған қатысты жүйелік ақпарат /etc/shadow файлында сақталады.

Меншікті UID GID анықтау үшін, id командасын пайдаланыңыз.

8.2.4.Файлдар және каталогтар.

Файл UNIX операциялық жүйесінің фундаментальды объктісі болып табылады. Барлық бағдарламалық жабдықтаулар файлдарда сақталады. UNIX ОЖ орналасқан компьютердің қатты дискісі, бөлім деп аталатын, бірнеше логикалық бөлімнен тұрады. Одан операциялық жүйелерді орналастыру үшін жасалатын, диск бөлімдерінен айырмашылығы UNIX ОЖ аймағында логикалық бөлімді көбінесе, слайс (slice) деп атайды. Слайстың орналасуы және өлшемі дискіні форматтау кезінде анықталады. UNIX – те олар тәуелсіз құрылғы ретінде болады, оларға қатынас құру, әртүрлі мәлімет жинақтауыштарға қатынас құру сияқты жүзеге асырылады. Слайстың бөлігі файлдық жүйені (filesystems) орналастыру үшін бөлінеді. Бір слайста тек қана бір файлдық жүйе орналаса алады. Диск тек қана бір слайстан тұра алады, бұл үлкен өлшемді файлдық жүйе жасауға мүмкіндік береді. UNIX өнімділігімен функциональдығымен және мәліметтерді сақтау сенімділігімен ерекшеленетін бірнеше файлдық жүйені пайдаланады: s5, ufs, nfs, vxfs және т.б. Файлдық жүйе кіретін слайстың жалпы құрылымын көрсету үшін, v System, UNIX операциялық жүйесінің s5 файлдық жүйесін қарастырамыз.

s5 файлдық жүйесі диск слайсын алады және үш негізгі құрауыштан тұрады:

-суперблок;

-индекстік дескрипторлар жиымы;

-мәліметтер блогы.

Суперблок – файлдық жүйе туралы жалпы ақпарат жинақталған:

-файлдық жүйенің түрі (тип)

-логикалық блоктағы файлдық жүйенің

суперблоктың өзін, дескрипторлар жиымын және мәліметтер блогын қоса алғандағы өлшемі;

-индекстік дескрипторлар жиымының өлшемі;

-орналастыруға болатын бос блоктар саны;

-дескрипторларды орналастыру үшін бос блоктар саны;

-логикалық блоктың өлшемі;

-бос дескрипторлардың нөмірлерінің тізімі;

-бос блоктардың адрестерінің тізімі;

Индекстік дескрипторлар массиві.

Индекстік дескрипторда (inode) файл туралы ақпарат, яғни файлдың метамәліметтер жинақталғаны. Әрбір файл, әрқайсысы бір, тура сол inode- ге нұсқайтын

тіпті файлдық жүйеде бірнеше атауға ие бола алатын, бір inode – мен байланысты. Индекстік дискриптор өрісі келесі ақпараттарға ие:

-файл түрі (типті) және қатынас құру құқығы;
-файлға сілтемелер саны, яғни файлдық жүйедегі файлдағы атау мөлшері;
-иесінің және топтың идентификиторы;
-байт пен файл өлшемі;
-арнайы файлдар үшін бұл өріске құрылғының кіші және үлкен нөмірі;
-файлға соңғы қатынас құру уақыты;
-файлдың соңғы модификацияланған уақыты;
-файл мәліметтері сақталатын дискілік блоктардың адрестерінің жиымы;

Дискілік блоктардың адрестерінің жиымында файл мәліметтерінің орналасуы туралы ақпара бар. Файл мәліметтері сақталған дискілік блоктар ретті орналаспауы да мүмкін болғандықтан, inode, осы файлға қатысты блоктарды адресін сақтау керек. Индекстік дискрипторда бұл ақпарат, әрбір элементтің де дискілік блоктың физикалық адресі бар, жиым түрінде сақталады. Ал файлдың логикалық блогының нөмірі жиым индексі болып табылады. Массивтңң 13 элементтен тұратын шектеулі өлшемі болады. Бірінші он элемент тікелей файл мәліметтері сақталатын блоктарды адрестейді, 11 элемент өз кезгінде мәліметтерді сақтайтын блоктар адресі бар, блокты адрестейді.

Он екінші элемент, блок адрестерін сақтаған , оның әрқайсысы файл мәліметтерін сақтаған блок адресін көрсететін, дискілік блокқа көрсетеді. Ол үшінші элемент –файл мәліметтерін сақтаған блок адресін табу үшін, үш қосымша блок қажет болғанда, үш еселенген жанама адрестер жасауға пайдаланылады.

Мұндай тәсіл, өлшемдері бірнеше байттан ондаған мегабайтқа дейін өзгеретін, салыстырмалы түрде шағындап тиянақталған өлшемге индекстік дескриптордың файлдармен жұмысын ұстап тұруға мүмкіндік береді .Салыстырмалы түрде шағын файлдар үшін(блок өлшемдері 1024 байтта болғанда 10 Кбайт- қа дейін) максимальды өнімділігті қамтамасыз ететін, тікелей индексациялау қолданылады. Өлшемдері 266 Кбайттан аспайтың (10 Кбайт+256*1024),қарапайым жанама адрестеу жеткілікті.

Соңғында, үш еселенген жанама адрестеуді пайдаланған сәтте, 16777216 (256*256*256) блокқа қатынас құруды қамтамасыз етуге болады.

Көптеген қазіргі операциялық жүйелер сияқты, UNIX ОЖ файлдар құрылымы ағашбейнесі түрінде ұйымдастырылған, және файлдар жүйесі деп аталады. Әрбір файлдың атауы бар, және атауы файлдар жүйесі құрылымындағы орынын анықтайды. Осы құрылымның түбірі "/" атауы бар түбірлік каталог болып табылады.

UNIX ОЖ тиесілі сипаттар:

- жүйеде ішкі құрылымы әртүрлі бірнеше файлдық жүйе қатысуы мүмкін;
- бұл жүйеге жататын файлдар жүйесі әртүрлі қондырғыларда орналастырылуы мүмкін;

Мынаны атап өту қажет, файл атауы – дискідегі мәлімет жиынтығы емес, файлдық жүйенің атрибуты болып табылады.

Жүйедегі әрбір файл,индекстік дескрипторда сақталған (inode), файлдың барлық сипаттамаларын сақтаушы , соңын ішінде файл мәліметі сақталған диксілік блоктар көрсеткішімен, (нұсқағышымен өзінің метамәліметімен (қосымша ақпараттар) байланысты.

Файл атауы – файлдық жүйеде оның метамәліметіне көрсеткіші болып табылады, ал бірақ, метамәліметте файл атауы көрсеткіш болмайды.

UNIX ОЖ функциональдық мақсаты және файлдармен санқилы операциялар орындауындағы операциялық жүйе әрекеті әртүрлі, бірнеше әртүрлі түрілі файл қолданыс табады.

Қарапайым файл(ordinary files) – мәлімет сақтаушы,файлдың неғұрлым жалпылама түрі.

ОЖ үшін, олар құрылымдалмаған мәлімет жиынтығы болып табылады.

Файл құрамын интеркретациялау – файлды өңдеуші бағдарлама арқылы жүзеге асады. Мұндай файлдарға мәтіндік файлдар, атқарушы бағдарламалар, бинарлық файлдар және т.б. жатады.

Каталог (directory). Каталогтар көмегімен файлдық жүйенің логикалық ағашы қалыпасады. Каталог- бұл өзінде орналасқан файлдардың атауын жинақтаған файл және операциялық жүйеге бұл файлдармен операция жүргізуге мүмкіндік беретің, қосымша ақпаратарға (мттамәліметер) көрсеткіштер жинақталған файл.

Құрылғының арнайы файлы (special file) физикалық құрылғыға қатынас құруды қамтамасыс етеді.

Символдық сілтемелер (symbolic link). Жоғарыда айтылып кткеніндей, кталогкта файлдар аитаулары және олардың метамәліметеріне көрсеткіштер бар. Метамәліметердің өздерінде файл атаулары да, ол атауға көрсеткіш те болмайды. Бұл файлдық жүйеде, бірғана файлға бірнеше атау иеленуіне мүмкіндік береді

Атаулар, метамәліметермен және сәйкесінше файлдың мәліметерімен тығыз байланысты, сонымен бірге, файл сияқты, файлдық жүйеде өзін қалай атайтындығына байланысыз бола алады .

Мұндай сілтемені In командасының көмегімен жасауға болады.

Файлдар және каталогтар: негізгі ұғымдар.

ОЖ файлдық жүйесін оқып үйренуге көшпестен бұрын, қабылданған кейбір сөздіктермен(термин) танысу қажет.

- Файлдық жүйенің орны (filesystem): дискінің, операциялық жүйеге, дискі блоктарына жылдам адрестеу және қатынас құруды жүзеге асыру мүмкіндігін беру үшін арнайы форматталған аймақ.
- Файл ағашы (tile system): біртұтас ағашбейнелі иерархиялық құрылымға логикалық біріктірілген, бір немесе бірнеше файлдар жүйесінің жиынтығы.
- Файл(tile): мәліметер сақталатын файлдық жүйенің ішіндегі атауы. Файл бос болуы да мүмкін (яғни, мәліметтер болмауы) бірақ та ол операциялық жүйе туралы белгілі бір ақпарат береді.
- Каталог (diretory): өзінде орналасқан файлдардың атаулары жинақталған файл.
- Ішкі каталог (subdirectory): басқа каталогтың ішінде орналасқан каталог. Құрамына ішкі каталогтар кіретін катлогтарды аналық каталогтар деп айтады.

Өзіне-өзі ата-ана болып табылатын тібірлік каталогты коспағанда, барлық каталогтарда аналық ішкі каталог болады.

- Файл атауы (tilename): файлдарды идентификациялау үшін жасалған символ тіркестері.
- Жол (pathname): "/" символымен біріктірілген, бір немесе одан көп файл атауларының тіркесі. Жол файлдың орнының, ағаш бейнелі файлдың құрылымының ішінде яғни файлдық ағаштың ішінде) орналасуын спецификациялайды.

Файл атаулары

UNIX ОЖ файлдармен жұмыс істеу үшін, олардың атауларын пайдаланбайды. Операциялық жүйеге қажетті барлық ақпарат, файлдың метамәліметтері, сәйкес дескриптордың реттік нөмірімен бірмәнді байланысты болатын, дескриптор жиымында орналасады. Файл атауы оның дескрипторының нөмірімен бірге файлдардың арнайы түрі - каталогтарда сақталады.

UNIX ОЖ файл атаулары ASCII символдарының комбиинацияланыынан тұрады.

Атау ұзындығы 255 символға дейін жетуі мүмкін (кейбір файлдың жүйелер ұзындықты 14 символға дейін шектейді).

Атауларда, арнайы міндеті бар символдарды пайдалануға рұқсат етілмейді.

: < > ' " ' S ! % Σ * ? / ( ) [ ]

(.) символынан басталатын файл атаулары, әдепкіде, ls командасымен шығарылмайтың, жасырынған (hidden) файлдарға жатады. Жасырынған файлдардың атауын шығару үшін , ls командасынан а опциясын пайдалану қажет.

Файл атауларында бас және кіші әріптерінде айырмашылық болады, сонықтан Test және test әртүрлі файлдарға қатысты болады.

Файл типтері

UNIX ОЖ файлдық жүйесі файлдың бірнеше типтерін қолдайды: әдетегі файлдар, символдық байланыстар, құрылғының арнайы файлдары.

Егер каталог мазмұны туралы ақпарат беретің, ls – l командасын орындаса, мынаны көруге болады: әрбір тіркестегі бірінші сөздің бірінші символы сәйкес файлдың түріне нұсқайды. Символдардың мәні төмендегідей - - қарапайым файлдар;

d- каталог;

l – символдық байланыс;

cb- құрылу файлдары.

Қарапайым файл - r w – r w.....

Каталог d r w x r w x ---.....

Символдық байланыстар l r w x r w x r w x ...

Құрылғы файлы c r w – r w ---- ....

Қарапайым файлдар

Қарапайым файлдар құрылымдандырылмаған байттар реттілігі бар. Осындай файлдармен жұмыс істейтін қосымша: оның мазмұнын және құрылымын анықтайды. Әдетте, осындай файлдарды келесі санаттардың біріне жатқызуға болады:

- Символдар жиынтығы бар мәтіндік. Мысалы, хат, есептер, shell интерфератоторы пайдаланатын командалық файлдар.
- Қандай да бір қосымшадағы мәліметердің мәтіндік және сандық жиынтығы бар. Мысалы, электрондық кестелер, мәтіндік поцессорлардың мәліметер қоры немесе құжаттары.
- Машиналық командалар және мәліметтер жинақталған екілік түрінде атқарылатын бағдарламалар . Мысалы, UNIX ОЖ командаларын орындаумен байланысты бағдарламалар немесе қосымшалардың бағдарламалары.

File командасын пайдалана отырып, файл түрін анықтауға болады. Команда бірнеше аргументтерді пайдалануға мүмкіндік береді, яғни бірнеше әртүрлі файлдардың түрін анықтауға мүмкіндік береді:

File аты _ [файл _ аты ...]

Каталогтар.

Каталогтар файлдық жүйенің иерархиялық құрылымын ұйымдастыруға арналған арнайы файл болып табылады.

Каталогтар файлдың файлдық жүйе ағашындағы қалпын анықтайды, өйткені файлдың өзінде өзінің орналасқан орны туралы ақпарат жоқ.

Каталогтарда да, қарапайым файлдардағы сияқты мәліметтер болады, бірақ та қарапайым файлдардан айырмашылығы, ядро бұл мәліметтердің құрылымына шектеу салады; каталогтарда әрбір файл үшін мынандай байлам түрінде болады: " индекстік дескриптор нөмірі – файл атауы":

- индекстік дескриптор нөмірі файл туралы барлық ақпаратар бар, индкстік кестесінің блок индексі ретінде пайдаланылады;
- файл атауы мәтіндік ақпарат(ASCII) болып табылады.

Каталогтарда бірнеше файлдарға қатысты , бірдей атаулар болмауы керек.

Әрбір каталогта бірінші атау ретінде "нүкте" (.) пайдаланылады, бұл каталогтың меншікті атауының синонимі, екінші атау ретінде "екі нүкте"(..) пайдаланылады, бұл жоғары тұрған ("аналық") кталог атауының синонимі.

Атауы "нүкте" символынан басталатың файлдар, жасырынған болып табылатындығын ескертуіміз керек, олар ls командасымен тек қана а опциясымен шығарылуы мүмкін болады.

Жаңа файл атауы қосылғанда, каталог өлшемі автоматты түрде үлкейеді, бірақ та, файл атауын жойғанда, каталог өлшемі кішіреймейді; жүйе ядросы катаолгтың босатылған бөлігін, қайтадан жасалған файл атауларының жазбасын (дәлірек айтқанда,"инекстік дескритор нөмірі – файл атауы") орналастыру үшін пайдаланады.

Жадыны басқару ішкі жүйесінің модулі жадының үдерістер арасында таратылуын қамтамасыз етеді.Егер барлық үдерістер үшін жады жеткіліксіз болса, ядро үдірістерді (белсенді) орындау үшін ресурстарды босата отырып, үдерістін бөлігін немесе бірнеше үдерістерді (көбінесе, жүйедегі қанайда бір оқиғаны күтуші (белсенді есес)) дискінің арнайы аймағана ("басқылау" аймағына) көшіреді.

Файлдық ішкі жүйе, дискілік жинақтауыштарда орналасқан мәліметерге және шалғай құрылғыларға қатынас құруға үйлестірілген интефеис қамтамасыз етеді. Ол файлдарды орналастыру және жою операцияларын орындайды, Файл мәліметерін жазу / оқу операцияларын орындайды, сол сияқты файлға қатынас құру құқын бақылайды.

Енгізу-шығару ішкі жүйесі,файлдық ішкі жүйенің және үдерістерді басқару ішкі жүйеснің і шалғай құрылғыларға қатынас құру сұраныстарын орындайды.

Ол сыртқы құрылғыларға қызмет көрсететін, ядроның арнайы бағдарламаларымен – құрылғы драйверімен өзара әрекет етеді.

8.3 Бақылау сұрақтары

8.3.1 Linux ОЖ функциялары қандай?
8.3.2 Linux ОЖ қандай құрауыштардан тұрады?
8.3.3 Linux ОЖ ядросы қандай құрауыштардан тұрады?
8.3.4 Linux ОЖ қолданушының идентификациясы қалай жүзеге асырылады?
8.3.5 Linux ОЖ файлдық жүйесі қандай құраушытардан тұрады?
8.3.6 Linux ОЖ файлдың қандай түрлері болады?

Изд. Лиц. № 0119644 от 21.08.2007
Подписано в печать 19.06.13. Формат 60х84/16.
Усл.печ.л. 4. Уч.-изд.л. 5.
Тираж 5. Заказ №195-13.
Цена договорная.

«Шыгыс Полиграф»
Усть-Каменогорск, ул. Ворошилова, 156