Olioperustainen Amiga EXEC

Jan Lindström jplindst@cs.helsinki.fi

Helsinki 25. huhtikuuta 1997

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos
Olio-ohjelmointi kevät 1997
Harjoitustyö 2
Ohjaaja: Juha Vihavainen

Sisältö

1 Johdanto

Olioperustaisuudesta on tullut eräs eniten toistettuja avainsanoja sekä teollisessa ohjelmistotuotannossa että ohjelmistotekniikan tutkimuksessa. Olioperustaisella ohjelmistokehityksellä tarkoitetaan systemaattista, koko ohjelmiston elinkaaren käsittävää prosessia, joka perustuu kaikissa vaiheissaan ohjelmistojen kuvaamiseen joukkona keskinäisessä vuorovaikutuksessa olevia olioita. Olioperustaisuus on yksi ohjelmointiparadigma, tapa kuvata erilaisten järjestelmien rakennetta ja toimintaa.

Olioperustaisen ohjelmistokehityksen idea on pelkistetysti se, että sovellusohjelmisto simuloi kyseiseen sovellukseen liittyviä konkreettisia tai abstrakteja tapahtumia ja olioita. Niinpä olioperustaisen ohjelmiston keskeinen piirre on, että se on samanrakenteinen sovelluksen käsitemaailman kanssa.

Käyttöjärjestelmä on eräs mahdollinen ohjelmisto, jonka suunnittelussa ja toteutuksessa voidaan käyttää olioperustaisuutta. Eräs käyttöjärjestelmä, jonka suunnittelussa on käytetty olioperustaisuutta on AmigaOS (Amiga Operating System). AmigaOS:n ytimen keskeiset osat on suunniteltu ja toteutettu käyttämällä perintää. AmigaOS:aa voi verrata parhaiten UNIX käyttöjärjestelmään, sillä AmigaOS sisältää käsitteellisellä tasolla useita samantapaisia osia. Yksityiskohdissa ja toteutuksessa on luonnollisesti suuriakin eroja, koska AmigaOS on alunperin tarkoitettu yhden käyttäjän käyttöjärjestelmäksi.

AmigaOS ei ole ainoa olioperustainen käyttöjärjestelmä, sillä olioperustaisia käyttöjärjestelmiä on tutkittu ja toteutettu useita. Eräs useiden käyttöjärjestelmien mikroytimenä käytetty olioperustainen käyttöjärjestelmä on Mach [ABB$^+$86, RBF$^+$89, RJO$^+$89], joka on siirretty myös Amiga:lle. Mach on myös atk-keskuksen Digital UNIX -koneiden käyttöjärjestelmän mikroytimenä. Muita olioperustaisia käyttöjärjestelmiä ovat esimerkiksi OOSY [Luu90], Apertos [Yok89], Muse [YTT89], ja SOS [SGH$^+$91].

Moniprosessorijärjestelmiin on myös toteutettu olioperustaisia käyttöjärjestelmiä. Eräs tällainen on Choises [CRJ87, CJM89, Cam91$^+$, Rus91]. Choises järjestelmän keskeisiä osia ovat prosessinhallinta [RJC88, Ley88], muistinhallinta [RuC89], laitehallinta [Kou91], tiedostojärjestelmä [Mad91] ja turvallisuus [CaM91]. Toinen esimerkki moniprosessorijärjestelmään toteutetusta olioperustaisesta käyttöjärjestelmästä on Elmwood [Mel87$^+$].

Hajautettuja olioperustaisia käyttöjärjestelmiä on esimerkiksi Spring [HaK93].

Soul [Sha91] on käyttöjärjestelmien toteuttamiseen tarkoitettu sovelluskehys. Sovelluskehys tarkoittaa olio-ohjelmoinnin yhteydessä kiinteästi toisiinsa liittyvien luokkien kokoelmaa, josta sopivasti täydentämällä ja kokoamalla saadaan valmis sovellus tai sen merkittävä osa.

Holloway [Hol91] on esittänyt erään mahdollisen oliomallinnuksen Amiga:n käyttöjärjestelmän ytimelle, mutta kuvauksessa ei ole käytetty mitään yleisiä menetelmiä.

Tämä artikkeli jakaantuu kahteen osaan, joiden tavoitteena on kuvata Amiga:n käyttöjärjestelmän ydin käyttäen OMT-menetelmää. Ensimmäisessä osassa esitetään ytimen keskeiset käsitteet ja ytimen osien oliomallinnus luokkakaavioiden avulla. Toisessa osassa kuvataan ytimen keskeisten osien yksityiskohtia.

2 Käyttöjärjestelmän oliomallinnus

Amiga:n käyttöjärjestelmä AmigaOS [Com86a, Com86b, Ami96, Mor87a, Mor87b] koostuu kolmesta keskeisestä osasta. Osat ovat moniajava ydin (Exec), tiedostojärjestelmä (AmigaDos) ja graafinen käyttöliittymä (Intuition, WorkBench).

Näistä kolmesta Exec on kaikkein olioperustaisin ja tämä artikkeli keskittyy sen perusrakenteen kuvaamiseen. Exec:n ydinrakenteen oliovastineet muodostavat yksijuurisen luokkahierarkian (liite 1). Jokainen Execin perusrakenne perii jonkin ylemmän tason yksinkertaisemmista luokista ja lisää uusia attribuutteja ja toimintoja perimäänsä luokkaan.

Oliomallinnuksen aluksi esitetään käyttöjärjestelmän käsitteet luetteleva ja määrittelevä mallisanasto. Tämän jälkeen ytimen osat kuvataan OMT-luokkakaavioina tarkemmin.

2.1 Mallisanasto

Device Laitteistoterminologiassa laitteisiin (devices) kuuluu levyasema, näppäimistö ja kaikki muut syöttö- tai tulostuslaitteet. Sovellusterminologiassa laite on käyttöliittymä tiettyyn syöttö- tai tulostuslaitteeseen.

Interrupt Keskeytyskäsittelijä on aliohjelma, joka palauttaa kontrollin systeemille päätyttyään. Keskeytyspalvelija on yksi tai useampia keskeytysrutiineja ketjutettuina peräkkäin.

IORequest Laajennettu viesti (Message), joka sisältää I/O-laitteelle kontrollitiedon ja datatietoa.

IOStdRequest Standardimallinen syöttö- ja tulostuskomento.

Library Tarjoaa organisoidun ryhmittelyn yhteenkuuluville palveluille, jotka muodostavat kirjaston. Tarjoaa palvelut kirjastojen luontiin ja ylläpitoon.

List Järjestetyn kaksisuuntaisen listan tunnussolmu.

MemList Vapaan muistialueen solmu.

MemHeader Vapaiden muistialuesolmujen listan tunnussolmu.

Message Tarjoaa tiedonvälityksen kahden prosessin välillä.

MinNode Muodostaa kahteen suutaan linkitetyn listan linkkikentät.

MinList Kaksisuuntaisen listan tunnussolmu.

MsgPort Portti, johon saapuvat viestit (Message) muodostavat jonon (List) tai johon lähetettävät viestit kirjoitetaan.

Node Kaksisuuntaisen listan solmu.

Resource Koordinoi jaettujen resurssien käyttöä.

Process Suoritettava ohjelma.

SemaphoreMessage Lukitusviesti jaettuun resurssiin. Jos resurssi on lukittu jää lukituspyyntö jonoon ja järjestelmä lähettää viestin lukinnan onnistumisesta lukitusta pyytäneelle prosessille.

SemaphoreRequest Jono jaettuun resurssiin odottavien prosessien viitteistä.

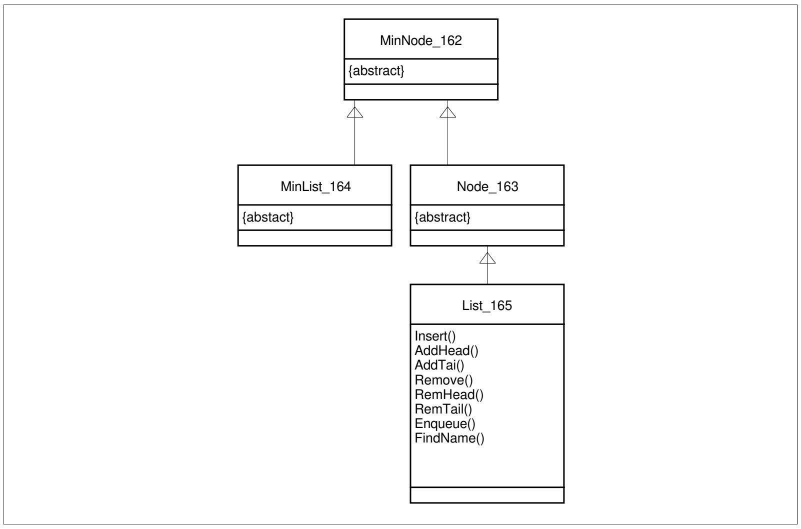
SignalSemaphore Lukituspyyntö jaettuun resurssiin. Jos resurssi on lukittu prosessi jää jonoon odottamaan resurssin vapautumista.

Task Suoritettavan ohjelma, joka ei sisällä syöttö- ja tulostuspalvelukutsuja.

Unit Tietty nimetty yksikkö viestiportista, josta on useampia ilmentymiä.

2.2 Luokkahierarkian juuri

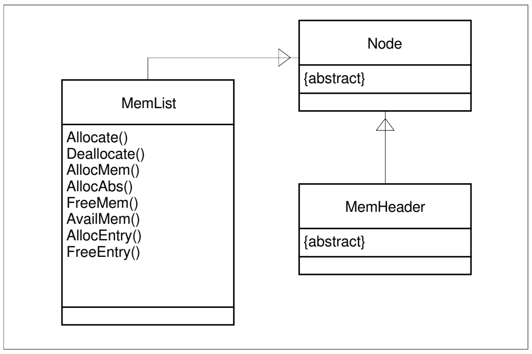
Koko luokkahierarkia alkaa MinNode-luokasta (kuva 1), joka on abstrakti luokka. MinNode-luokka sisältää attribuutteinaan kaksi linkkikenttää, joiden avulla kaikki MinNode-luokan perivät luokat voidaan linkittää listaksi. MinList-luokka perii MinNode-luokan ja toteuttaa sovellusohjelmoijalle järjestämättömän listan perusominaisuudet. Node-luokka perii MinList-luokan ja toteuttaa listalle järjestysominaisuuden. Node-luokka on yliluokkana lähes kaikille Exec:n perusrakenteille. List-luokka on kaksisuuntaisen listan tunnussolmu, joka perii Node-luokan (kappale 3.2).



Kuva 1: Luokkahierarkian juuri.

2.3 Muistinhallinta

MemList-luokka ja MemHeader-luokka (kuva 2) perivät Node-luokan. Luokat tarjoavat palvelut muistinhallintaan (kappale 3.7).

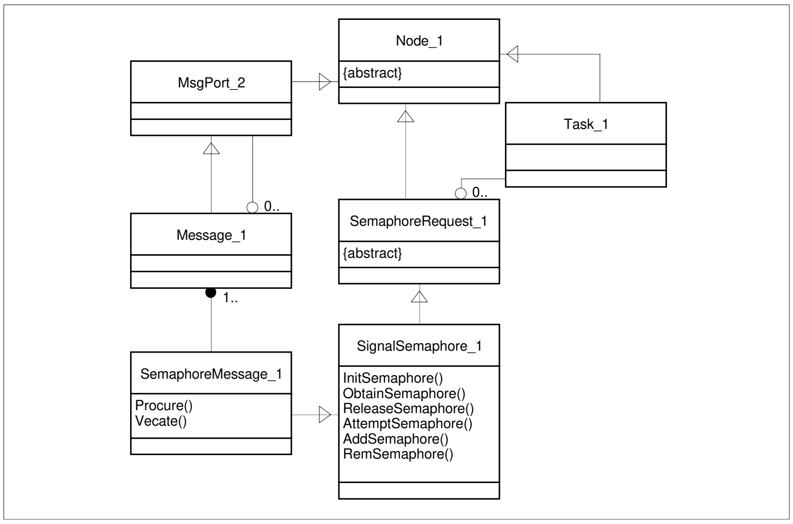


Kuva 2: Muistinhallinnan luokat.

2.4 Semaforit

SemaphoreRequest-luokka (kuva 3) perii Node-luokan tarjoten semaforien käsittelyyn yleiset abstraktit palvelut. SemaphoreRequest-luokan ilmentymä liittyy aina yhteen Task-luokan ilmentymään ja yhteen Task-luokan ilmentymään voi liittyä useita SemaphoreRequest-luokan ilmentymiä.

SignalSemaphore-luokka perii SemaphoreRequest-luokan ja huolehtii signaalimuotoisten semaforien palveluista. SemaphoreMessage-luokka perii SignalSemaphoreluokan ja tarjoaa viestimuotoisten samaforien palvelut. SemaphoreMessage-luokkaan liittyy useita Message-luokan ilmentymiä ja Message-luokan ilmentymä viittaa yhteen SemaphoreMessage-luokan ilmentymään.

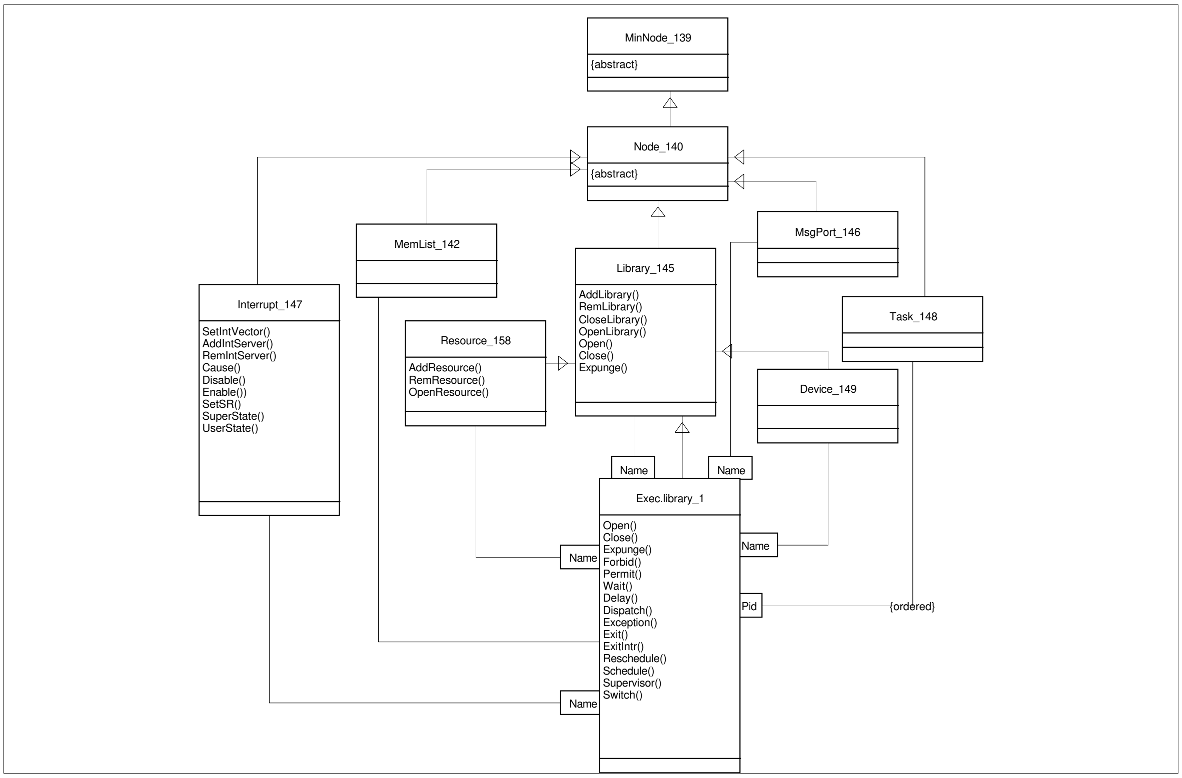


Kuva 3: Semaforiluokat.

2.5 Käyttöjärjestelmän ydin

Library-luokka on yksi Execin tärkeimmistä luokista (kappale 3.3). Library-luokka perii Node-luokan. Jokaisesta Library-luokan perineestä kirjastoluokan ilmentymästä voi olla vain yksi ilmentymä olemassa. Resource-luokka perii Library-luokan ja tarjoaa yhteiskäyttöisten resurssien hallinnan. Interrupt-luokka perii Node-luokan ja tarjoaa palvelut keskeytysten hallintaan (kappale 3.8).

Exec.library -luokka (kuva 4) perii Library-luokan ja toteuttaa käyttöjärjestelmän ytimen tärkeimmät toiminnot. Luokasta on aina olemassa vain yksi ilmentymä, jonka nimi on exec.library (kappale 3.1). Exec:iin liittyy nimellä tunnistettavasti yhdestä useampia MsgPort-luokan, Library-luokan, Resource-luokan, Interrupt-luokan ja Device-luokan ilmentymiä. Exec:iin liittyy prosessitunnisteella tunnistettavasti ja prioriteetin mukaan järjestetysti useita Task-luokan ilmentymiä. Lisäksi Exec:iin liittyy yksi MemList-luokan ilmentymä.



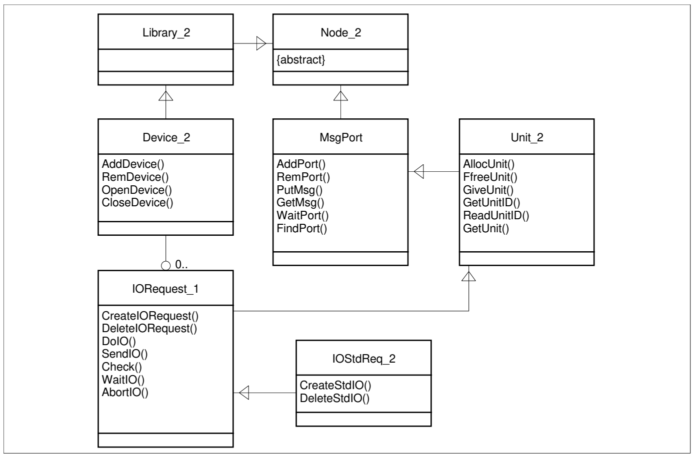
Kuva 4: Kirjastot ja exec-kirjasto.

2.6 Syöttö- ja tulostuslaitteet

Device-luokka (kuva 5) perii Library-luokan ja tarjoaa palvelut laitteiden käsittelyyn ja toteutukseen (kappale 3.6).

MsgPort-luokka perii Node-luokan ja toteuttaa viestiporttien käsittelyn (kappale 3.5). Unit-luokka perii MsgPort-luokan ja samoin Message-luokka perii MsgPort-luokan ja toteuttaa prosessien välisen viestinvälityksen (kappale 3.5). MsgPort-luokkaan liittyy yhdestä useampia Message-luokan ilmentymiä ja Message-luokan ilmentymä kuuluu aina yhteen MsgPort-luokan ilmentymään.

IORequest-luokka perii Unit-luokan ja tarjoaa syöttö- ja tulostuskomentojen lähetyksen laitteistolle (kappale 3.6). IOStdReq-luokka perii IORequest-luokan ja tarjoaa standardimuotoisten syöttö- ja tulostuskomentojen välityksen laitteistolle. IORequest-luokan ilmentymä liittyy aina yhteen Device-luokan ilmentymään ja Device-luokan ilmentymään voi liittyä useita IORequest-luokan ilmentymiä. IORequest-luokan ilmentymään liittyy joukko Message-luokan ilmentymiä ja Message-luokan ilmentymä liittyy aina yhteen IORequest-luokan ilmentymään.



Kuva 5: Laitteidenhallinnan luokat.

3 Käyttöjärjestelmän ytimen osat

3.1 Exec

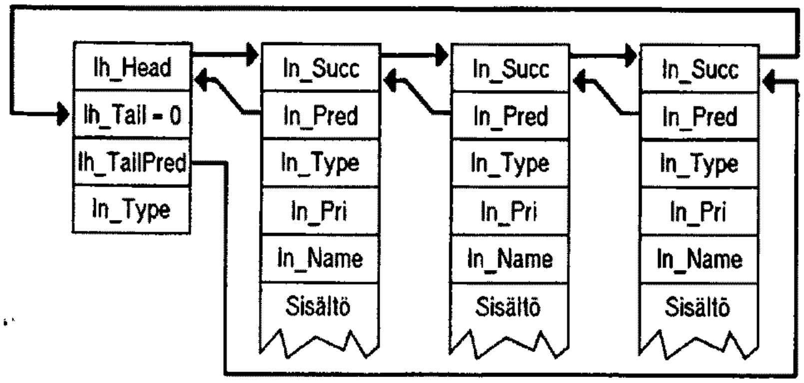
Exec:n yksi perustehtävistä on rutiini- eli aliohjelmakirjastona toimiminen muiden kirjastojen tapaan. Tämän lisäksi Exec huolehtii muun muassa seuraavista tehtävistä:

- Koneen alustus kylmäkäynnistyksestä sekä kaikkien laajennuskorttien ja laitteiden *liittäminen* (configuration) järjestelmään.
- Muistin varaaminen ja vapauttaminen prosessien tarpeiden mukaan.
- Moniajon aikaansaaminen vaihtelemalla nopeasti ajettavana olevaa ohjelmaa prosessien tarpeiden ja vapaan prosessoriajan mukaan.
- Viestiporttien hallinta ja eri prosessien välisten viestiliikenteen käsittely.
- Syöttö- ja tulostuskomentojen välittäminen eri laitteille.
- Keskeytysten hallinta ja ohjaaminen oikealle keskeytysrutiinille.

Exec huolehtii tehtävistään pitämällä kaksisuuntaisissa listoissa tai jonoissa viitteitä varsinaisten olioiden ilmentymiin.

3.2 Listat ja jonot

Exec ketjuttaa kirjastot, syöttö- ja tulostuslaitteet, ajettavat ohjelmat, vapaat muistialueet ja muut resurssit omiksi listoikseen. Lista koostuu *tunnussolmusta* (header node) ja kaksisuuntaisesti ketjutetuista *alkioista* (nodes). Tunnussolmu sisältää viitteet ensimmäiseen ja viimeiseen listan alkioon (kuva 6). Jonot ovat listoja, joiden alkiot on järjestetty prioriteettiensa mukaiseen järjestykseen suurimman prioriteetin alkiot ensimmäisiksi.



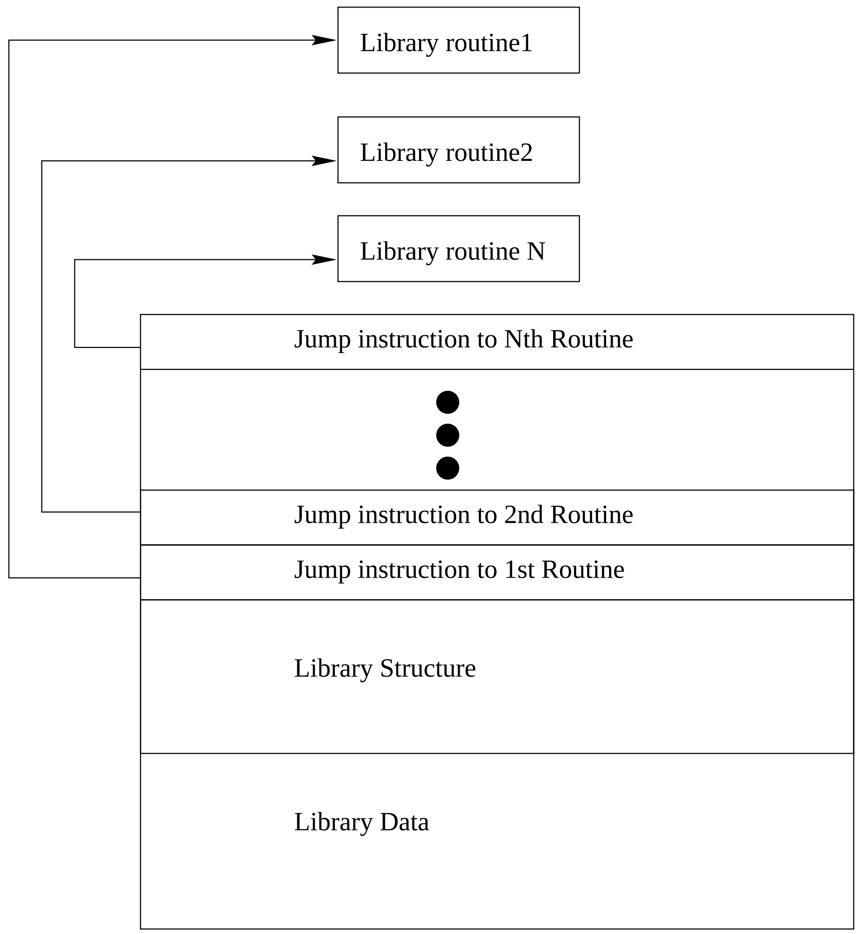
Kuva 6: Listan rakenne.

3.3 Kirjastot

Käyttöjärjestelmä koostuu useista erillisistä *kirjastoista* (library), jotka jokainen sisältävät omaan alaansa liittyviä rutiineja eli käyttäjän kutsuttavia aliohjelmia. Kirjastot voivat olla joko toisistaan täysin riippumattomia tai hyödyntää toinen toistensa rutiineja halutun toiminnan aikaansaamiseksi. Useimmat ylemmät kirjastot riippuvat exec-kirjaston olemassaolosta.

Nimitys *ajonaikainen kirjasto* (run time library) tarkoittaa sitä, että kirjaston sisältämät rutiinit ovat varsinaisen ohjelman ulkopuolella ja ohjelman käytettävissä nimenomaan ohjelman ajon aikana. Ajonaikasten kirjastojen vastakohta ovat *linkitettävät kirjastot* (linker library), jotka tulevat ohjelman käännös- ja linkitysvaiheessa kiinteäksi osaksi ohjelmaa itseään ja joita ei sen vuoksi tarvita erikseen ohjelmaa ajettaessa. Amigan käyttöjärjestelmän kirjastot ovat ajonaikaisia.

Jokainen kirjasto sisältää joukon hyppykäskyjä, systeemikirjaston alkion (Node) ja datasegmentin. Hyppykäskyjoukko on kuusitavuisten hyppykäskyjen muodostama taulukko, jonka välityksellä kirjaston rutiineja kutsutaan. Kirjaston lopussa on kirjaston mahdollisesti tarvitsemat tietorakenteet sekä useissa tapauksissa myös itse kirjastorutiinit (kuva 7).



Kuva 7: Kirjaston rakenne.

Jokaiseen kirjastoon kuuluu vähintään kolme rutiinia, jotka ovat tarpeen kirjaston avaamista, sulkemista ja muistista poistamista varten. Kirjastoon tarvittavat rutiinit ovat omat Open-, Close- ja Expunge- rutiinit. Open-rutiini ajetaan aina, kun joku avaa kirjaston. Close-rutiini ajetaan kirjastoa suljettaessa ja Expunge-rutiinin tehtävänä puolestaan on vapauttaa kirjaston tarvitsemat resurssit siinä tapauksessa, että kirjasto päätetään poistaa muistista kaikkien käyttäjien suljettua sen.

3.4 Prosessit ja moniajo

Moniajo on yksi tärkeimmistä ominaisuuksista, jonka Exec tarjoaa. Moniajon toteutus Exec:ssä vastaa periaatteellisella tasolla UNIX:n vastaavaa. Jokaista Amigan muistiin ladattua, ajossa olevaa ohjelmaa nimitetään *tehtäväksi* (task). Tiedostojärjestelmää (dos.library) tarvitsevat tehtävät ovat *prosesseja* (process), jotka perivät Task-luokan. Tehtävä eroaa prosessista siinä, että tehtävä ei voi suorittaa syöttö- tai tulostuskomentoja. Exec:n kannalta tehtävät ja prosessit ovat samanarvoisia ja molempia käsitellään samalla tavalla. Jatkossa käytetään prosessi nimitystä kuvaamaan sekä tehtäviä että prosesseja.

Prosesseja voi olla useita, vaikka Amigassa on vain yksi prosessori. Exec huolehtii prosessoriajan jakamisesta eri prosessien kesken, jolloin syntyy vaikutelma useista samanaikaisesti toimivista ohjelmista.

Prosessit on jaettu kahteen eri jonoon. Toisessa jonossa ovat ne prosessit, jotka odottavat suoritusvuoroa. Toisessa jonossa taas ovat ne prosessit, jotka odottavat jonkin *tapahtuman* (event) valmistumista. Suurin osa prosesseista on jatkuvasti odottamassa jotain tapahtumaa, kuten näppäimen painallusta tai levyoperaation päättymistä.

Exec:n moniajo syntyy *limittämällä* (multiplexing) prosessoria prosessien välillä. Jokaisella prosessilla on prioriteetti, joka määrää sen, miten keskusyksikköaika jakautuu prosessien kesken. Korkeamman prioriteetin prosessit saavat aina aikaa ennen matalamman prioriteetin prosesseja. Jokainen prosessi toimii kuten *virtuaaliterminaali* (virtual terminal), jolla on täysi kontrolli prosessoriin ja *laitteistoon* (hardware).

Prosessien vaihto on normaalisti *keskeyttävää* (preemptive). Ajossa oleva prosessi voi menettää prosessorin lähes millä hetkellä tahansa, jolloin se korvataan toisella kiireellisemmällä prosessilla. Myöhemmin, kun prosessi pääsee uudelleen prosessorille, se jatkaa kohdasta johon jäätiin. Prosesseja on mahdollista ajaa myös *yhteistyömoniajossa* (cooperative multitasking). Tällainen ajotapa on normaalisti varattu systeemin tietorakenteiden käsittelyyn.

Prosessien priorisoidun skeduloinnin lisäksi samanprioriteettisten prosessien välille syntyy *osituskäyttö* (time-slicing). Tällöin jokaista prosessia ajetaan tietyn *aikaviipaleen* (time-slice) ajan, jolloin prosessien välille muodostuu *vuorottelu* (round robin). Jos prosessi ylittää aikaviipaleensa, systeemi keskeyttää sen ja jatkaa toisen prosessin suorittamista.

Prosessin *vaihto* (rescheduling) voi siis tapahtua kolmesta eri syystä. Prosessi itse jää odottamaa jotakin tapahtumaa tai keskeytysrutiini on signaloinut jotain korkeamman prioriteetin prosessia tai prosessi on käyttänyt loppuun aikaviipaleensa ja seuraava saman prioriteetin prosessi otetaan ajettavaksi.

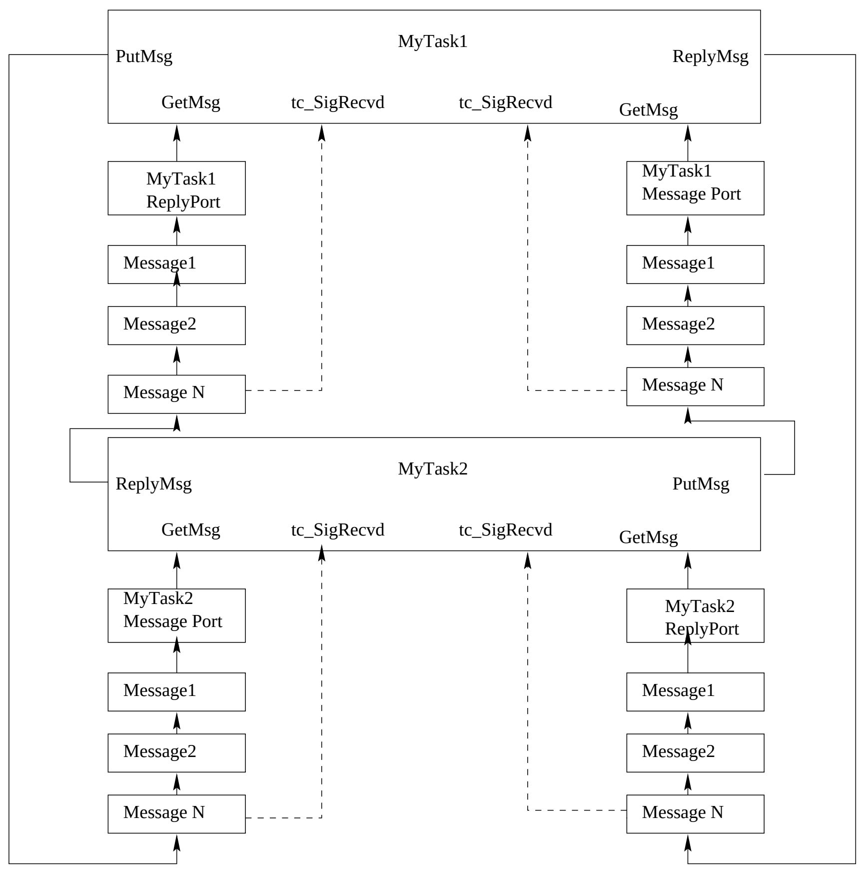
Exec ajaa kaikkia prosesseja *käyttäjätilassa* (user state). *Suojatussa tilassa* (supervisor state) ajetaan *prosessorivirheet* (processor trap) ja tietyt keskeytykset. Tästä syystä kaikki käyttöjärjestelmän osat voidaan korvata vaihtoehtoisilla toteutuksilla jopa ajonaikaisesti. Toisaalta ongelmana on se, että virheellisesti toimivan ohjelma voi sekoittaa käyttöjärjestelmän tietorakeiteiden sisällön aiheuttaen koneen kaatumisen.

3.5 Viestit ja portit

Moniajokäyttöjärjestelmän on pystyttävä välittämään nopeasti ja helposti tietoa prosessista toiseen riippumatta siitä, mitä kyseiset prosessit ovat tekemässä. Amigan laitteistotason syöttö ja tulostus perustuu *viestien* (messages) käyttöön. Syöttö- ja tulostuslaitteelle lähetetään komentoviesti, jonka laite suorittaa pääohjelmasta erillisenä ja ilmoittaa pääohjelmalle komennon suorituksen päättymisestä.

Execin käsittelemä viesti on yksinkertainen muistialue, jonka alussa on Message-luokan ilmentymä. Message-luokka on perinyt Node-luokan, jonka avulla viestit ketjutetaan listaksi (kuva 8).

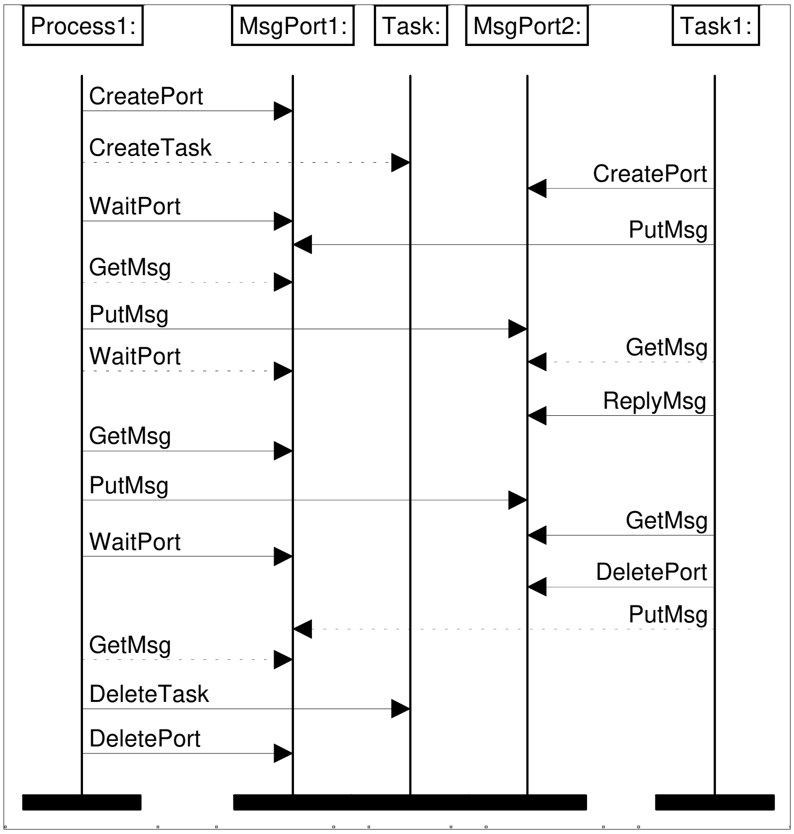
Viestit lähetetään aina vastaanottavan prosessin *viestiporttiin* (message port). Portissa sisääntulevat viestit asetetaan jonoon FIFO-jonon tapaan. Portit ovat viestien välitystä varten varattuja tietorakenteita, jotka sisältävät tietoja vastaanottavasta prosessista sekä listan porttiin saapuneista viesteistä.



Kuva 8: Kahden prosessin välinen viestinvälitys.

Jonossa olevat viestit ovat aina viitteitä. Tehokkuussyistä viestien kopiointia ei suoriteta. Kahden prosessin välillä kulkeva viesti on väliaikainen lupa vastaanottavalle prosessille käyttää osaa lähettävän prosessin muistialuetta. Esimerkiksi jos prosessi

A lähettää viestin prosessille B, viesti on prosessin A muistialueella. Prosessi A ei voi kuitenkaan käsitellä viestin muistialuetta ennen kuin viestiin on *vastattu* (reply). Kuvassa 9 on esimerkki prosessin ja tehtävän välisestä viestinvälityksestä esitettynä tapahtumakaaviolla.



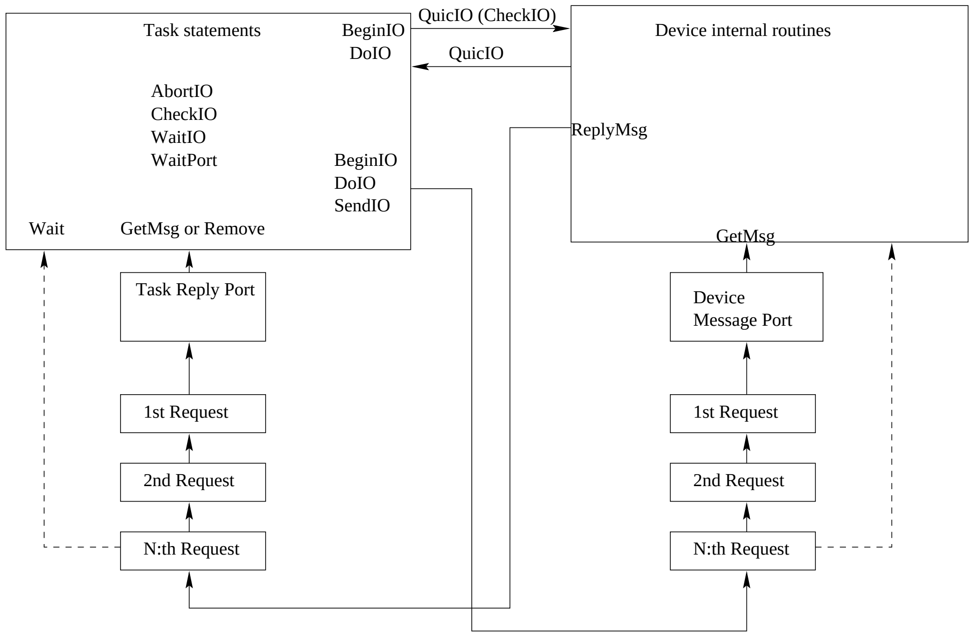
Kuva 9: Esimerkki viestinvälityksestä.

Viestiportit ovat *kohtaamispisteitä* (rendezvous points), missä viestejä kerätään. Portti voi koostua useista odottavista viesteistä eri lähteistä. Kun viesti saapuu porttiin, viesti lisätään vastaanottavan portin viestijonon loppuun ja ennalta määritelty toimenpide aktivoidaan. Toimenpide voi olla tyhjä tai se voi aiheuttaa ennalta määritellyn signaalin tai ohjelmallisen keskeytyksen.

3.6 Syöttö- ja tulostuslaitteet

Olennainen osan Amigan käyttöjärjestelmää muodostavat erilaiset *syöttö- ja tulostuslaitteet* (devices), joiden kautta suurin osa syöttö- ja tulostustoiminnoista tapahtuu. Osa syöttö- ja tulostuslaitteista (esimerkiksi levyasema) perustuu suoraan tietyn laitteen käyttöön, osa on toteutettu kokonaan ohjelmallisesti. Riippumatta syöttö-ja tulostuslaitteiden takana olevista toteutuksista niitä kaikkia ohjataan samojen peruskomentojen avulla.

Suurta osaa syöttö- ja tulostuslaitteista hoitaa oma ohjelmansa, mutta osa yksinkertaisimmista laitteista toimii kokonaan keskeytysten ja käyttäjäkutsujen varassa (esimerkiksi timer.device). Riippumatta laitteen sisäisestä toiminnasta käskyt annetaan aina syöttö- ja tulostusrutiineja ja vakiokomentoja käyttäen (kuva 10). Monet laitteet ymmärtävät myös toimintojensa vaatimia erikoiskäskyjä, jotka on määritelty erikseen kunkin laitteen palveluissa.



Kuva 10: Laite ja prosessi.

Syöttö- ja tulostuskomennot perustuvat aiemmin käsiteltyjen viestien ja porttien käyttöön. Jokaisella syöttö- ja tulostuslaitteella on oma viestiportti, johon sovellusohjelmat lähettävät laitteen suoritettaviksi tarkoitetut komennot. Viestejä ei lähetetä käyttäen viesteistä tuttuja aliohjelmia, vaan erityisiä syöttö- ja tulostuskutsuja, jotka edelleen kutsuvat halutun laitteen sisäisiä palveluja.

Kaikki komennot ja parametrit laitteille annetaan IORequest-luokan ilmentymän avulla. IORequest-luokka on perinyt Node-luokan, jonka avulla laite voi ketjuttaa sille saapuvat komennot listaksi ja käsitellä ne yksitellen.

3.7 Muistinhallinta

Amigan kaltaisessa systeemissä ei voi olla kiinteitä muistiosoitteita eikä rajatun kokoisia taulukoita. Muistiavaruuskin on yhtä poikkeusta lukuunottamatta dynaaminen ja eri tietorakenteet sijoittuvat muistissa minne milloinkin.

Amigassa ainoa kiinteä tunnettu muistiosoite on AbsExecBase, joka sijaitsee muistin alapäässä osoitteessa $00000004. Tämän osoitteen avulla on mahdollista saada selville kaikkien käyttöjärjestelmärutiinien ja tietorakenteiden osoitteet avaamalla kirjastoja ja kutsumalla niiden rutiineja.

Exec, Amigan käyttöjärjestelmän alin taso, huolehtii muistin jakamisesta eri tarpeisiin. Erilaiset ohjelmat pyytävät muistia kulloistenkin vaatimustensa mukaan ja Exec luovuttaa niille vapaata muistia sieltä, mistä sitä milloinkin sattuu olemaan. Exec pitää tiedossaan ainoastaan vapaat muistialueet. Sovellusohjelmien tulee huolehtia varaamansa muistin vapauttamisesta. Poiketen muista vastaavista käyttöjärjestelmistä (kuten Unix) Exec ei toteuta valmiiksi muistinsuojausta ja virtuaalimuistia.

3.8 Keskeytykset

Keskeytys (interrupt) tarkoittaa ajettavan ohjelman toiminnan keskeyttämistä ja prosessorin luovuttamista erillisen keskeytysrutiinin haltuun. Keskeytysrutiinin ajon jälkeen käynnistetään keskeytynyt sovellus jälleen täsmälleen samasta paikasta, jossa se oli keskeytetty.

Keskeytyksillä on erittäin suuri merkitys Amigan kaltaisessa tietokoneessa, jossa tapahtuu samanaikaisesti useita hyvinkin erilaisia asioita. Myös moniajo perustuu osaltaan keskeytysten käyttöön. Käyttöjärjestelmän lisäksi myös käyttäjän ohjelmat saattavat joskus tarvita keskeytysten suoraa käyttöä. Keskeytykset liittyvät yleensä läheisesti laitteistoon ja sen hallintaan.

Amigassa keskeytykset jaetaan kahteen pääryhmään sen perusteella, voiko kyseistä keskeytystä olla hoitamassa useita erillisiä, peräkkäin ketjutettuja rutiineja, vai ainoastaan yksi rutiini kerrallaan. Jos rutiineja voi olla useita samanaikaisesti, puhutaan *keskeytyspalvelijoista* (interrupt servers), jos keskeytykseen voi liittyä vain yksi rutiini, puhutaan *keskeytyskäsittelijöistä* (interrupt handlers).

4 Yhteenveto

Tämä artikkeli ja aiemmat tutkimukset osoittavat, että olioperustaisuutta voidaan käyttää monimutkaistenkin sovellusten mallinnuksessa. Käyttöjärjestelmä on tietokoneen tärkein ja ehkä monimutkaisin sovellus. Olioperustaisuuden avulla käyttöjärjestelmästä muodostuu ylläpidettävä ja laajennettava kokonaisuus.

Olioperustaisen ohjelmistokehityksen jokaisessa vaiheessa voidaan valita suoritetaanko vaihe olioperustaisesti vai tavanomaisin menetelmin. Siksi olioperustainen ohjelmistokehitys ei aseta rajoituksia ohjelmointikielelle.

Amiga:n käyttöjärjestelmän ydin Exec, on hyvä kuinka olioperustaisesti suunniteltu järjestelmä voidaan toteuttaa rakenteisella ohjelmointikielellä säilyttäen osa olioperustaisuudesta.

Kaikki Amiga:n käyttöjärjestelmän osat kuten tiedostojärjestelmä, graafinen käyttöliittymä ja sovellusohjelmat käyttävät Exec:n palveluita. Exec on kiinteä, tehokas, joustava, luotettava ja laajennettava. Ja kuten tämä artikkeli osoittaa, Exec:n rakenne on olioperustainen.

Lähteet

[ABB+86] Accetta, M.J., Baron, R., Bolosky, W., Golub, D., Rashid, R., Tevanian, R.F., Young, M.W., Mach: A New Kernel Foundation for UNIX Development. Proc. of the Summer USENIX. July, 1986.

[Ami96] Amiga Technologies GmbH., Amiga Developer CD v1.1, Schatzruhe, Germany, 1996.

[CaM91] Campbell, R., Madany, P., Considerations of Persistence and Security in Choises an Object-Oriented Operating System. Technical report UIUCDCS-R-91-1670, March 1991, University of Illinois at Urbana-Champaign.

[Cam91+] Campbell, R., Islam, N., Johnson, R., Kougiouris, P., Madany, P., Choices, Frameworks and Refinement. Technical report UIUCDCS-R-91-1712, December 1991, University of Illinois at Urbana-Champaign.

[CJM89] Campbell, R., Johnston, G., Madany, P., Principles of Object-Oriented Operating System Design. Technical report UIUCDCS-R-89-1510, April 1989, University of Illinois at Urbana-Champaign.

[Com86a] Commodore Inc., Amiga ROM Kernel Reference Manual: Exec Addison-Wesley, 1986.

[Com86b] Commodore Inc., Amiga ROM Kernel Reference Manual: Libraries and Devices Addison-Wesley, 1986.

[CRJ87] Campbell, R., Russo, V., Johnston, G., The Design of a Multiprocessor Operating System. In USENIX C++ Conference, USENIX Association, November 1987.

[HaK93] Hamilton, G., Kougiouris, P., The Sprint Nucleus: A Microkernel for Objects. SMLI TR-93-14, April 1993, Sun Microsystems Laboratories, Inc.

[Hol91] Holloway, T. The Object-Oriented Amiga Exec BYTE, January 1991, 329–334.

[Ley88] Leyens, D., A Choises implementation of the universal scheduling system. Master's thesis, 1988, University of Illinois at Urbana-Champaign.

[Luu90] Luursema, J., OOSY,een Object Oriented Operating System. Universiteit Twente, December 1990.

[Kou91] Kougiouris, P., A Device Management Framework for an Object-Oriented Operating System Technical report UIUCDCS-R-91-1707, May 1991, University of Illinois at Urbana-Champaign.

[Mad91] Madany, P., An Object-Oriented Framework for File Systems. Technical report UIUCDCS-R-91-1751, June 1991, University of Illinois at Urbana-Champaign.

[Mel87+] Mellor-Crummey, J., LeBlanc, T.J., Crowl, L.A., Gafter, N.M., Dibble, P.C., Elmwood - An Object-Oriented Multiprocessor Operating System. Technical report 226, September 1987, University of Rochester.

[Mor87a] Mortimore, E.P., Amiga Programmer's Handbook Volume I SYBEX, 1987.

[Mor87b] Mortimore, E.P., Amiga Programmer's Handbook Volume II SYBEX, 1987.

[RBF+89] Rashid, R., Baron, R., Forin, A., Golub, D., Jones, M., Julin, D., Orr, D., Sanzi, R., Mach: A Foundation for Open Systems Proc. of the Second Workshop on Workstation Operation Systems September, 1989.

[RJC88] Russo, V., Johnston, G., Campbell, R., Process Management and Exception Handling in Multiprocessor Operating Systems using Object-Oriented Design Techniques. Proc. of OOPSLA'88 Conf., San Diego, California, September 25–30, 1988, p248-258.

[RJO+89] Rashid, R., Julin, D., Orr, D., Sanzi, R., Baron, R., Forin, A., Golub, D., Jones, M., Mach: A System Software Kernel Proc. of the 34th Computer Society Int. Conf., February, 1989.

[RuC89] Russo, V., Campbell, R., Virtual Memory and Backing Storage Management in Multiprocessor Operating Systems using Object-Oriented Design Techniques. Proc. of OOPSLA'89 Conf., New Orleans, Louisiana, October 1–6, 1989, p267–278.

[Rus91] Russo, V., An Object-Oriented Operating System. Technical report UIUCDCS-R-91-1670, March 1991, University of Illinois at Urbana-Champaign.

[Sha91] Shapiro, M., Soul: An object-oriented OS framework for object support. Workshop on Operating Systems for Nineties and Beyond, Dagstuhl Castle, Germany, July 1991.

[SGH+91] Shapiro, M., Gourhant, Y., Habert, S., Mosseri, L., Ruffin, M., Valot, C., SOS: An Object-Oriented Operating System - Assessment and Perspectives. Computing System 2(4), 287–337,1989.

[Yok89] Yokote, Y., The Apertos Reflective Operating System. The Concept and Its Implementation. Proc. of OOPSLA'92 Conf., Vancouver, British Columbia, October 18–22, 1992, p414–434.

[YTT89] Yokote, Y., Teraoka, F., Toroko, M., A Reflective Architecture for an Object-Oriented Distributed Operating System. Proc. of ECOOP'89 Conf., University of Nottingham, July 10–14, 1988, p89–106.