Tartu Ülikool
Matemaatika-informaatikateaduskond
Arvutiteaduse instituut

Mitmeprotsessilise operatsioonisüsteemi

*Meth*OS

dokumentatsioon

Aine “Operatsioonisüsteemide ehitus”
projektitöö

Autorid:
*Meth*OS Grupp
Juhendajad:
Varmo Vene
Meelis Roos

Tartu 2003

Sissejuhatus

MethOS (MitmETHegumiline OperatsiooniSüsteem) on OSKiti-põhine alglaetav operatsioonisüsteem, milles on realiseeritud väljatõrjuva mitmetegumilisusega protsessihaldur. MethOS sisaldab planeerijat, kestprogrammi ja näidisprogramme, mille abil on võimalik demonstreerida ajajaotust ning protsessidevahelist suhtlust.

Kokkuvõte MethOSist

MethOS sisaldab järgnevaid komponente ja võimalusi:

- OSKitil põhinev x86 platvormi alglaetav operatsioonisüsteem
- Tekstirežiim ja klaviatuurisisend
- Mälufailisüsteem
- Toetab virtuaalmälu ja saalefaili
- Tuuma lõimedel põhinev protsessihaldus
  - Kahetasemeline tagasisidega järjekord
  - ELF-vormingus programmide laadimine
- Protsessidevahelise side võimalus teadete abil
- Kasutaja tasemel on tuuma funktsionaalsus kättesaadav süsteemifunktsioonide abil
- Hulk lisatud näidisprogramme

Algne projekt

Algselt oli plaanis koostada PC-tüüpi arvutile minimalistlik operatsioonisüsteem, millel töötab väljatõrjuva mitmetegumilisusega protsessihaldur. Protsessihaldur suudab luua protsesse ning garanteerida nende vahel õiglase ajajaotuse. Protsessidel on ka omavahelise andmevahetuse võimalus, mis aitab neil oma tööd koordineerida.

MethOS tugineb OSKiti teekidele ja töötab teoreetiliselt kõigil OSKiti poolt toetatud platvormidel. MethOSi väljatöötamine ja testimine toimub x86-arhitektuuril. Operatsioonisüsteemi tuuma saab kõvakettalt laadida *multiboot*-standardile vastava laaduriga, milleks sobivad näiteks LILO2 ja GRUB3.

Kogu vajaliku riistvaratoetuse võtab MethOS OSKiti vahenditest. Samuti kasutab MethOS OSKiti vahendeid sisendi ning väljundiga tegelemisel, (virtuaal)mälu haldamisel ja failisüsteemi kasutamisel.

MethOS kasutab protsessoriaja jaotamisel väljatõrjuvat planeerijat. Kõigil protsessidel on dünaamilised prioriteedid. Töö iseloomu järgi liigitatakse protsessid kaheks – interaktiivseteks ja taustprotsessideks. Liigitus ei ole automaatne, seega käivitatakse kõik protsessid interaktiivsetena. Taustprotsessid võivad töö alguses protsessisüsteemile teatada, et nad soovivad töötada taustal.

Planeerijal on mõlemat liiki protsesside jaoks eraldi järjekord. Interaktiivsete protsesside järjekorras toimub käivitatava protsessi valik nn. *Round-Robin* algoritmi järgi, kusjuures interaktiivsetele protsessidele ei anta üle 80% protsessoriajast. Taustprogrammide järjekorrast valib järgmise protsessi väljatõrjuv *Shortest Job First* (SJF) algoritm. Ajahinnang antakse statistiliselt vastavalt protsessi eelnevale käitumisele. Protsessi "nälgimist" välditakse tema prioriteedi tõstmisega ooteaja kasvades.

Protsessid saavad teineteisele saata lihtsaid teateid, mis kogunevad vastuvõtva protsessi "postkasti".

MethOS võimaldab rakendusprogrammidele minimaalse hulga C-keele funktsioone. Primitiivne sisend-väljund on realiseeritud süsteemifunktsioonidena. Loomulikult on saadaval ka komplekt andmetöötlusfunktsioone. Eraldi on välja toodud protsessihalduse funktsioonid.

Näidistena on MethOSi juures nii interaktiivseid kui taustprogramme. Interaktiivsetest programmidest on kõige olulisem kestprogramm, mis võimaldab teisi näidisprogramme käivitada. Teiste näidisprogrammide eesmärgiks on pakkuda protsessihalduse testimiseks vajalikku protsessisegu.

Protsessihaldus on ehitatud OSKiti lõimeteegi (*POSIX threads*) baasil ja lihtsa protsessiteegi (*Simple Process*) laiendusena. Planeerija realiseeritakse lõimeteegi vastavaid laiendusi (*CPU Inheritance Framework*) kasutades. See võimaldab võtta üle lõimeteegi ülesanded protsessoriaja planeerimisel. Selle abil laiendatakse OSKiti lihtsat protsessihaldussüsteemi eelpoolkirjeldatud funktsionaalsusega.

Süsteemifunktsioonid C-keele teegi ja rakenduste jaoks realiseeritakse MethOSi tuumas. Väljakutsed toimuvad tarkvaralise katkestuse genereerimise teel.

Muudatused

Üldjoontes on lõpptulemus algselt planeerituga sarnane, aga seoses sellega, et OSKiti *CPU Inheritance Framework*i ei ole kaua uuendatud ning ta ei ühildu ülejäänud OSKitiga, on tehniline

lahendus erinev.

Taustprogrammide järjekord on sarnaselt interaktiivsete protsesside järjekorrale *Round-Robin* (RR) tüüpi algselt planeeritud *Shortest Job First*i asemel. Protsessidel prioriteete ei ole, aga nendega on seotud muutuja, kus on kirjas kui palju aega RR igas tsüklis talle andma peab. Interaktiivsete protsesside puhul on see väärtus staatiline (20 ms) ja taustprotsesside puhul dünaamiline. Ta muutub vastavalt sellele, kas eelmise RR tsükli ajal soovis protsess rohkem või vähem aega kui talle anti, aga mitte vähem kui 10 ms ja mitte suurem kui 40 ms. Protsessi käivitamiseks taustal, tuleb ta kestprogrammis käivitada vastava parameetriga (&).

Protsessidevaheliseks suhtlemiseks ei ole igal protsessil oma postkasti vaid on üks ühine, millest iga protsess saab vaadata, kas ja kellelt on talle sõnumeid. Selle jaoks on sõnumil kaasas siht- ja saatjaprotsessi identifikaator.

Protsessihaldus

Protsesside loomine ja haldus

MethOSi protsessihaldus võimaldab operatsioonisüsteemil käivitada samaaegselt mitut erinevat programmi. ELF-vormingus programme laetakse mälufailisüsteemist. Programme saab kirjutada C-keeles. Keskkonnaks sobib Linux-operatsioonisüsteemiga arvuti. Programmide kompileerimisel tuleb kasutada teatud võtmeid ja lisada MethOSi ja OSKiti teeke.

Protsesse identifitseeritakse protsessinumbri järgi. Protsessi loomisel eraldatakse sellele kirjete massiivist (*process pool*) struktuur ning tagastatakse (sessiooni jooksul) unikaalne number (piiratud 16-bitise täisarvuga). Loodud protsessi sisse saab laadida programmifaili, mis seejärel käivitatakse.
Igal protsessil on opereerimiseks enda mälualad (hallatud OSKit *Simple Process* teegi abil). Protsess koosneb kontroll-lõimest ja protsessilõimest. Kontroll-lõim vastutab protsessilõime käivitamise ja selle töö lõpul protsessi ressursside vabastamise eest. Protsessilõimes töötab laetud programm. Mõlemad on tuuma tasemel loodud lõimed.

Protsess võib jääda ootama teise protsessi töö lõppu. See on kasulik protsesside loomisel. Protsesse saab ka nime järgi otsida, sellel juhul tagastatakse protsessi number. Ülejäänud operatsioonid tegelevad planeerija parameetrite muutmise ja protsessiinfo väljastamisega.

Programmeerimisliides:

```
mth_process_init, mth_process_create, mth_process_loadelf,
```

`mth_process_join, mth_process_destroy, mth_process_self, mth_process_list`

Planeerija

Planeerija kasutab kahetasemelist tagasisidega järjekorda, kus protsessid jaotuvad kaheks: interaktiivseteks ja taustprotsessideks. Kõik loodavad protsessid on vaikimisi interaktiivsed. Nende ajakvant on 20 millisekundit ja see ei muutu. Interaktiivsete protsesside järjekorras liiguvad protsessid *Round-Robin* algoritmi järgi. Interaktiivseid protsesse võtab lühiajaline planeerija töösse maksimaalselt 80% ajast.

Kui programm on oma olemuselt nn „deemon"-tüüpi programm, mis töötab tagaplaanil, võib ta liikuda taustprotsesside järjekorda. Taustprotsesse vaikimisi ei looda. Esialgne ajakvant on 20 ms, kuid see muutub vastavalt protsessi vajadustele dünaamiliselt. Kui protsess tõrjutakse välja, suurendatakse tema ajakvanti 10 ms võrra (maksimaalselt 40 ms). Vastupidiselt, kui protsess annab endale eraldatud protsessoriaja tagasi, kasutades selleks eraldi süsteemifunktsiooni, vähendatakse selle ajakvanti 10 ms võrra (minimaalselt 10 ms).

Sellise algoritmiga garanteeritakse aeg-ajalt „ärkavate" protsesside optimaalne ajakasutus – nad ei pea kasutama hõivatud ootamist protsessidevahelise teate või mõne tingimuse täitumiseni. Kehtivad ka piirangud. Nimelt ei saa taustprotsessid kasutada klaviatuurisisendit – sellised süsteemifunktsioonid jätavad lõime ootama esiplaanile tõstmist.

Protsesse saab järjekordade vahel vabalt liigutada. Kui taustprotsess muuta interaktiivseks, muudetakse tema ajakvant taas 20 ms suuruseks. Planeerija teeb ka statistikat protsesside ajakasutuse kohta.

Kirjeldatud algoritmi realiseerimiseks kirjutasime OSKiti *pthreads*-teegile uue planeerija, laiendades olemasoleva teegi võimalusi mitmetasemelise tagasisidega järjekorra kasutamiseks.

Programmeerimisliides:

`mth_process_settype, mth_process_gettype, mth_process_yield`

Protsessidevaheline side

MethOS kasutab väga lihtsat protsessidevahelise andmevahetuse tehnikat. Protsessid saavad omavahel suhelda teadete teel. Praeguses realisatsioonis on teateks maksimaalselt 100 märgi pikkune sõne. Kui protsess soovib saata teadet, otsib ta vastava meetodi abil nime järgi välja

sihtprotsessi numbri. Seejärel koostab ta enda sõnumi ning saadab selle ära. Teade läheb süsteemi suurde teatejärjekorda.

Kui protsess soovib teada, kas talle on teade saadetud, kontrollitakse järjekorda ning tagastatakse esimene teade, mille adressaadiks on teadet küsinud protsess.

Programmeerimisliides:
`mth_message_init, mth_message_send, mth_message_receive`

Süsteemifunktsioonid

Kasutajataseme programmidel on tuuma funktsionaalsusele ligipääs süsteemifunktsioonide kaudu. OSKiti *Simple Process* teegi abil realiseeritud süsteemifunktsioonide mehhanism kasutab väljakutsete tegemiseks tarkvaralisi katkestusi, argumendid edastatakse mälutabeli kaudu.
Kasutajaprogrammidele on saadaval mitu gruppi funktsioone:

- sisend-väljund konsoolil ja mälufailisüsteemis (printf, gets, puts, fopen, fclose, feof)
- protsessihaldus (mth_process_*)
- sõnumite saatmine (mth_message_*)
- mõned lihtsas C-teegis realiseerimata funktsioonid (atoi, srand, rand, time)
- muud funktsioonid (sleep)

Programmeerimisliides:
mth_syscall_init

Näiteprogrammid ja kestprogramm

Kestprogramm 'kest'

MethOSi tuum laeb pärast käivitumist mällu esimese protsessina kestprogrammi 'kest'. Kesta ülesandeks on teiste programmide käivitamine ja minimaalne protsesside haldamine. Kest võimaldab programme laadida nii interaktiivsete- kui taustprotsessidena. Taustale laadimiseks tuleb programmi nime lõppu lisada sümbol &. Taustale laetud protsessi esiplaanile toomiseks on kestprogrammil käsk 'yh' (ühine). Veel tunneb kestprogramm käsku 'v2lju', mis lubab kestprogrammil väljuda.

Kest kontrollib ka programmi olemasolu (tuuma „mugavusfunktsiooni“ mth_fileexists abil) ja ei püüa käivitada olematut programmi. Kõiki sisendeid peale eelpoolmainitute tõlgendatakse programminimedena ning püütakse laadida vastavat programmi. Kõige esimesest kestprogrammist väljudes läheb kontroll tagasi tuumale, mis lõpetab operatsioonisüsteemi töö.

Arvu arvamise komplekt 'arvamine'

See on programmikomplekt, mis koosneb programmidest 'arvamine-server', 'arvamine-klient' ja 'arvamine-stopp'. Tegemist on protsessidevahelist sidet demonstreeriva paketiga, mis koosneb juhuarve genereerivast serverist ning kasutajaliidesega kliendist, mis lubab mängijal ära arvata arvu vahemikus 1 – 100.

Server ootab sõnumit 'palun arvu', millele vastab genereeritud arvuga. Kui sisendteateks on 'aitab', lõpetab server töö ja väljub. Muud väljumisvõimalust serveril pole, seega on otstarbekas see käivitada taustprotsessina. Klient otsib käivitudes süsteemist serverit. Kui see leitakse, saadetakse päring 'palun arvu'. Saadud arvu lubatakse kasutajal ära arvata, andes vihjeid 'minu arv suurem' ja 'minu arv väiksem'. Serveri töö lõpetab programm 'arvamine-stopp', mis saadab serveril teate 'aitab'.

Abiprogramm 'abi'

Selle programmi eesmärgiks on olemasolevate programmide nimekirja väljastamine. Ta kirjutab ekraanile programmide nimekirja ning ka abiinfo kesta lisakäskude kohta.

Näidisprogramm 'sekund'

See programm kirjutab ekraanile sada korda kirja 'sekund', jättes kordade vahele ühe sekundi.

Protsessiinfo 'pr-info' , 'pr-taustal'

Töötavate protsesside kohta jagavad infot need kaks programmi. Esimene neist väljastab selle vaid ühel korral, teine jääb tsüklisse ning väljastab protsessiinfot iga sekundi järel. Protsessiinfo koosneb protsessi numbrist, nimest, tema lõimede numbritest, protsessi tüübist (interaktiivne või taustprotsess), ajakvandist, protsessikasutuse protsendist ja üldse kasutatud protsessoriajast.

Süsteemifunktsioonide testkomplekt 'sf-test'

Üheteistkümnest testist koosnev komplekt näitab sisend- ja väljundfunktsioonide tööd nii ekraani kui faili puhul.

Protsessidevahelise side näide 's6num-test'

Programm küsib kasutajalt sisendit ning saadab selle sõnumina iseendale.

Lõpmatu tsükli näide 'igavene'

Programm, mis koosneb ühest käsust – while (1);

Meeskond

Dan Bogdanov – protsessihaldus

Kadri Hendla – planeerimisalgoritm, dokumentatsioon

Marko Jõemets – kestprogramm

Liina Kamm – planeerimisalgoritm, dokumentatsioon

Reimo Luik - näidisprogrammid