HUUG
EurOpen

NJSZT

HUNIX' 92

konferencia előadásai

Gödöllő
1992. szeptember 28-30

A KONFERENCIA SZPONZORAI:

Controll-Elektronikai és Számítástechnikai Rt.
Digital Equipment Corporation (Magyarország) Kft.
Hewlett Packard Kft.
IBM Magyarország Kft.
IIF Koordinációs Iroda
International Computers Limited (ICL-Hungary)
IQSOFT Intelligens Software Rt.
Magyar Távközlési Vállalat (MATÁV)
Microsystem Rt.
Miniszterelnöki Hivatal
ONYX Szoftverház Kft.
Unisys Sysland Kft.

Tartalomjegyzék

IV.szekció:**Adatbázis kezelés UNIX környezetben**

V.szekció: **UNIX alapú alkalmazások**

VI.szekció: **A 90-es évek stratégiái**

I.szekció: **U N I X JELENE és JÖVŐJE**

Jelen és jövő a UNIX bázisú kommunikációban

Horváth Nándor, MTA SZTAKI
<horvath@sztaki.hu>

Az 1970-es évek elején, amikor Amerikában elkezdték kiépíteni az első nagy távolságú adathálózatukat, valószínűleg nem gondolták volna, hogy 20 év alatt ilyen rohamos fejlődésen mennek keresztül a számítógép hálózatok. A 70-es évek ARPA hálózata (már akkor) 56 kbit/sec sebességű vonalakat használt, és a hallatlan sikere miatt kellett megszüntetni: olyan sokan kezdték használni, hogy ez a vonalkapacitás nem bírta el az ugrásszerűen megnövekedett forgalmat. Azóta Amerikában több száz Mbit/sec sebességű vonalakat használnak, ami húsz év alatt az adatátviteli kapacitásban 8000-szeres növekedést jelent. Ugyanilyen nagy arányú volt a növekedés a bekapcsolt host-ok számában, és most már a különféle technológiára alapuló hálózatok az egész világot behálózzák.

A UNIX operációs rendszerű gépek között alapvetően két fajta hálózati technológia terjedt el: a UUCP és a TCP/IP alapú hálózatok. A különböző felhasználói programok is főleg ezeket támogatják, természetesen a mai modern UNIX-os gépekkel más hálózatokba is bekapcsolódhatunk. A TCP/IP volt az alapja az ARPA hálózatnak, és ezt használják az ezt kiváltott újabb hálózatok is. A TCP/IP-re alapuló hálózatok összessége alkotja a mai Internet-et, amelynek most már Magyarország is tagja. A UUCP rendszert a Bell Laboratórium keszítette 1975-ben, az akkor még túl drágának tartott TCP/IP technológia helyettesítésére, és a kisebb forgalmú helyeken még ma is sikerrel alkalmazzák, mert a UUCP a legmostohább viszonyok között is (például kapcsolt telefonvonal) képes működni. Sajnos azonban akinek csak UUCP-je van, nem tudja elérni az egyébként TCP/IP-vel elérhető összes szolgáltatást.

A UUCP és a TCP/IP főbb jellemzői

A UUCP működését tekintve tulajdonképpen egy file-transfer protokoll, amely leírja, hogy hogyan kell a távoli géppel kapcsolatba kerülni, és file-okat átvinni egyik helyről a másikra. A file-transfer protokollnak három változata van, igazodva a fizikai közeg sajátosságaihoz. Az **f** protokoll X.25 hálózat fölött működik optimálisan, a **g** protokoll képes kijavítani a vonalon (telefon) keletkezett átviteli hibákat, a **t** protokoll pedig TCP/IP fölött használható. Valamilyen információ átvitele ebben a rendszerben két lépésben történik: az egyik a tényleges adatfile átvitele, a másik egy parancsfile átvitele, amely előírja, hogy a távoli rendszeren mit kell ezzel a file-lal csinálni.

Látjuk tehát, hogy ilyen módon csak egy batch jellegű feldolgozás oldható meg, és közvetlen on-line kapcsolatra a UUCP rendszer nem biztosít lehetőséget. Sok szolgáltatás, amely nem igényel közvetlen kapcsolatot, azonban így is jól használható, mint például az elektronikus levelezés, file-transfer, NetNews, stb.

A TCP/IP-re alapuló hálózatok alapvetően csomagkapcsolt hálózatok, ezért sok hasonlóságot lehet felfedezni köztük, és a Magyarországon is már elterjedten használt X.25 hálózattal. A TCP/IP hálózatok akkor működnek jól, ha a két állomás között valamilyen állandó jellegű összeköttetés van, például bérelt vonal. A csomópontokban nagyon gyakran egy speciális kapcsológépet (router) használnak. Ez a kapcsológép biztosítja az adatcsomagok útvonalának optimális kiválasztását, a különböző vonalak közötti továbbítását. Az egyes számítógépek ezek után általában egy-egy ilyen router-en keresztül kapcsolódnak a nagy távolságú hálózatokhoz.

Az állandó összeköttetés itt már lehetővé tesz on-line kapcsolatokat is, például a saját terminálunkról beléphetünk egy távoli gépbe, stb. A közvetlen kapcsolat miatt még az alapvetően batch jellegű szolgáltatásoknál is, mint például a levelezés, sokkal gyorsabban tudunk információt továbbítani. A UUCP rendszerekben szokásos néhány óra helyett már 1-2 percen belül eljuthat a címzetthez a levelünk.

A következőkben ismertetni fogom a jelenlegi magyarországi lehetőségeket és az Amerikában, Nyugat-Európában jelenleg elérhető szolgáltatásokat, amelyek meghonosítása Magyarországon épp csak elkezdődött. A következő egy-két év feladata lesz, hogy megteremtsük azt a hazai alaphálózatot, infrastruktúrát, amely széles körben elérhetővé teszi mindenki számára ezeket a szolgáltatásokat.

Jelenlegi magyarországi lehetőségek

Jelenleg Magyarországon főleg az elektronikus levelezés terjedt el, a felhasználók az egyéb szolgáltatásokat szinte kizárólag csak ennek a közvetítésével érhetik el. Az elektronikus levelezés vagy a nyilvános X.25 hálózaton keresztül, vagy kapcsolt telefonvonalon érhető el. UNIX alapú rendszerek általában a UUCP protokollt használják, MS-DOS PC-ken pedig főleg az IIF intézményekben jól ismert ELLA rendszert használhatjuk, de létezik a UUCP programcsomagnak egy public-domain változata, amelyik PC-ken is fut.

A levelezésen kívül UNIX-os gépekről elérhetjük a NetNews szolgáltatást is. A NetNews rendszerben a számítógépek minden résztvevő által egyformán értelmezett cimkékkel (úgynevezett news-csoportok) megjelölt üzeneteket cserélnek egymással. A levelekkel ellentétben egy ilyen üzenetnek (cikknek) nincs meghatározott címzettje, hanem megkapja a NetNews rendszerbe bekapcsolt összes gép, amely az adott news-csoportra "előfizetett". A news-cso-

portok témakörük szerint hierarchikus rendszert alkotnak. A fő news-csoportok: comp, misc, sci, soc, talk, news, rec, alt, gnu, biz. Ez alatt szűkebb és szűkebb témakörök találhatók, például: comp.ibm.pc.binaries, comp.sys.hp.

Egyre több intézmény vásárol UNIX alapú munkaállomásokat, server-eket. Ezeken a gépeken már azonnal rendelkezésre állnak a TCP/IP alapszolgáltatások, és így a lokális hálózaton lévő gépek között már most is használhatunk remote login-t, Network File System-et, stb. A következő lépés, hogy ezeket a lokális hálózatokat egymással összekössük, sőt, megteremtsük a nemzetközi összeköttetésekre is a lehetőséget.

A közelmúltban kezdődött el a hazai TCP/IP alapú nagy távolságú hálózat tervezése, és remélhetőleg rövidesen rendelkezésre állnak a nagy sebességű (64 kbit/s) nemzetközi vonalak is. Véleményem szerint körülbelül fél éven belül hazánkban is széles körben elérhetővé válnak majd az Internet szolgáltatások.

A továbbiakban áttekintem a nyugaton már működő szolgáltatásokat, amelyek Magyarországon is a bekapcsolódó intézmények számára rövidesen elérhetővé válnak.

Telnet

A Telnet a leggyakrabban használt Internet szolgáltatás, ha egy távoli géppel szeretnénk on-line kapcsolatba kerülni. A Telnet egy hibamentes adatátviteli csatornát biztosít a felhasználó számára, és a kapcsolat felépülte után úgy viselkedik, mintha a távoli gép egy terminálja előtt ülnénk.

Használata: `telnet <internet cím>`
Például: `telnet nic.ddn.mil` vagy `telnet 192.6.11.5`

Sok szolgáltatást már a Telnet segítségével is elérhetünk. Itt most csak néhány példát említek meg, ezeken kívül számtalan egyéb lehetőség van a Telnet alapú szolgáltatások igénybevételére.

- Nagyobb amerikai és európai könyvtárak katalógusában, adatbázisában on-line kereshetünk.

- Sok Interneten keresztül elérhető intézmény üzemeltet valamilyen adatbázist, amely a kutatóinak, dolgozóinak legfontosabb adatait (pl. e-mail cím) tartalmazza.

- A legtöbb nagy (fizetős, vagy ingyenes) adatbázis (pl. DIALOG) lekérdezhető, elérhető az Interneten keresztül is.

- A RIPE (Európai Kutatói IP Hálózatok Szervezete) információs rendszerében elérhetők a legfontosabb Internet dokumentációk, az európai

backbone hálózatról (EBONE) készült anyagok, lekérdezhető a RIPE adatbázis, amely az európai IP hálózatok fontosabb adatait tartalmazza, stb.

FTP, Anonymous FTP

Az FTP (File Transfer Protocol) az elsődleges eszköz, amellyel az Interneten file-okat vihetünk át egyik gépről a másikra. Ehhez általában az szükséges, hogy mindkét gépen rendelkezzünk azonosítóval, amivel beléphetünk a rendszerbe. Sok információ, például a public-domain software-ek, mindenki által elérhetők kell legyenek, ezért létrehoztak egy speciális azonosítót, amellyel mindenki olvashatja ezeket a nyilvános file-okat. Ezt a módszert nevezzük Anonymous FTP-nek. Az FTP elindításakor meg kell mondanunk, hogy mely géppel szeretnénk felépíteni a kapcsolatot, pl: `ftp ftp.sztaki.hu`. Ez után a rendszer megkérdezi, hogy milyen user-azonosítót használjon a távoli gépen, és mi ott a jelszavunk. Ha az Anonymous FTP-t engedélyezték, felhasználói azonosítóként `anonymous`-t kell megadnunk, és a jelszó általában a saját nevünk, vagy e-mail címünk. Miután a saját, vagy anonymous azonosítóval beléptünk, elindíthatjuk a file-átvitelt.

Ezek után felmerül a kérdés, hogy egy adott programcsomagot (ennek is a legfrissebb változatát!) melyik gépen érhetjük el. A sok ezer FTP szolgáltatógép között sokáig keresgélhetnénk, míg a keresett anyagot megtaláljuk, ezért rendszeres időközönként nyilvánosságra hoznak listákat, amelyek a legnagyobb anonymous FTP server-eket tartalmazzák, és hogy ott milyen témakörben találhatók programcsomagok. Így már csak néhány server-t kell végignéznünk. Még ez sem igazán kényelmes, ezért kifejlesztették az Archie (ld. alább) adatbázisokat, amelyekben sokkal kényelmesebben kereshetünk, és a válasz is lényegesen pontosabb, így például kiválaszthatjuk a hozzánk legközelebbi gépet, és onnan hozzuk át a file-t, hogy a hálózat terhelése minél kisebb legyen.

Finger

A Finger parancs segítségével lekérdezhetjük, hogy egy távoli gépen ki van az adott pillanatban belogonolva (első példa), illetve részletesen is lekérdezhetjük egy adott felhasználó adatait (második példa).

Például: `finger @fserv.kfki.hu` vagy `finger horvath@sztaki.hu`

Ping

A Ping parancs segítségével ellenőrizhetjük, hogy egy távoli gép a hálózaton keresztül elérhető-e. Segítségével arról is képet kaphatunk, hogy a hálózati kapcsolat milyen gyors, illetve milyen nagy a hálózat terhelése.

Például:

```
% ping aearn
PING aearn (192.78.2.1): 56 data bytes at 1.0000 second intervals
64 bytes from 192.78.2.1: icmp_seq=0. time=245. ms
64 bytes from 192.78.2.1: icmp_seq=1. time=239. ms
64 bytes from 192.78.2.1: icmp_seq=2. time=243. ms
^C
----aearn PING Statistics----
3 packets transmitted, 3 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 239/242/245
```

Talk

Néha valaminek a megbeszélése elektronikus levélben nehézkes, és valódi interaktív kapcsolatra lenne szükség. Ilyenkor használhatjuk a Talk szolgáltatást, amelynek formája: **talk user@hostname**. Ilyenkor a távoli felhasználó kap egy üzenetet a termináljára (feltéve, hogy éppen be van logonolva), hogy "beszélgetni" szeretnénk vele. Ha elfogadja a hívásunkat, a két terminál összekapcsolódik, és mindketten látják, amit a partnerük gépel.

Whois

A Whois adatbázist a NIC (Network Information Center) és a RIPE üzemelteti. A 'whois' parancs segítségével kereshetünk ezekben az adatbázisokban, amelyek a regisztrált Internet domain neveket, Internet címeket, rendszeradminisztrátorok adatait, stb. tartalmazzák. A RIPE adatbázis európai adatokra specializálódott, a NIC-nél lévő általánosabb, de nem olyan részletes. Néhány példa:

```
% whois brendan
```

— default server: NIC, keresi a Brendan nevű személyt.

```
Kehoe, Brendan (BK59)                brendan@cs.widener.edu
   Widener University
   Department of Science
   17th & Walnut Streets
   Chester, PA 19013
   (215) 499-4528

   Record last updated on 09-Jun-92.
```

```
% whois -h whois.ripe.net bme.hu
```

— a 'whois.ripe.net' server-en a bme.hu domain név után érdeklődünk.

```
domain:     bme.hu
descr:      Technical University of Budapest
admin-c:    Laszlo Fekete
zone-c:     Laszlo Fekete
remarks:    MX-only
changed:    dfk@cwi.nl 920309
```

```
person:     Laszlo Fekete
address:    Technical University of Budapest
address:    Centre of Information Systems
address:    1111 Budapest
address:    Muegyetem rkp. 9. III.e. 310
address:    Hungary
phone:      361 1 812 174
fax-no:     361 1 665 711
e-mail:     h487FEK@ella.hu
```

A fent említett két legfontosabb server-en kívül sok más intézmény is üzemeltet Whois server-t, ezek lekérdezése a második példához hasonló módon történhet.

Archie

Az Archie server-eket azért hozták létre, hogy megkönnyítsék a felhasználóknak a különböző public domain softwar-ek utáni keresést. Ehhez kilistázták a világon elérhető anonymous FTP server-ek tartalmát, és az így keletkezett hatalmas listát betöltötték egy adatbázisba. Az így keletkezett adatbázis az Interneten elosztva több számítógépen is lekérdezhető, hogy mindenki a hozzá legközelebbi adatbázist használhassa. Az adatbázis egy kliens software-en keresztül érhető el, de ennek hiányában Telnet-tet is használhatunk. Ilyenkor először fel kell hívnunk a kiválasztott szolgáltató gépet, és be kell logonolnunk **archie** felhasználóként. Ez után rendelkezésünkre áll egy elég részletes használati utasítás (Help), illetve a **prog** *<software-név>* paranccsal rögtön kereshetünk is az adatbázisban. Ha volt találat, a rendszer kilistázza, hogy pontosan melyik anonymous FTP server-en található a keresett file, és kiírja a teljes directory bejegyzést is (dátum, hossz, stb.).

Jelenleg az Archie server-eket az alábbi gépeken érhetjük el:

```
archie.rutgers.edu  128.6.18.15    (Rutgers University)
archie.unl.edu      129.93.1.14    (University of Nebraska in Lincoln)
archie.ans.net      147.224.1.2    (ANS archie server)
archie.mcgill.ca    132.206.2.3    (Canada server;original archie site)
archie.au           139.130.4.6    (Australian server)
archie.funet.fi     128.214.6.100  (European server in Finland)
archie.doc.ic.ac.uk 146.169.11.3   (UK/England server)
archiecs.huji.ac.il 132.65.6.15    (Israel server)
```

Egy példa az Archie használatára:

```
login: archie
SunOS Release 4.1.2 (ARCHIE) #3: Sat Feb 15 15:09:08 EST 1992
              Welcome to the ARCHIE server at SURAnet

archie> prog archie
# matches / % database searched:     3 / 100%

Host capella.eetech.mcgill.ca    (132.206.1.17)
Last updated 17:07 23 Aug 1992

    Location: /wuarchive/usenet/comp.sources.amiga/volume89/util
```

```
      FILE       rw-rw-r--        5054  Mar 16  1989   archie.1.Z

Host world.std.com   (192.74.137.5)
Last updated 06:21 21 Aug 1992

    Location: /src/amiga/util
      FILE       rw-r--r--        5054  Feb 27  1990   archie.1.Z

Host wuarchive.wustl.edu   (128.252.135.4)
Last updated 03:12 21 Aug 1992

    Location: /usenet/comp.sources.amiga/volume89/util
      FILE       rw-rw-r--        5054  Mar 16  1989   archie.1.Z
```

Wais

Egy kezdő Internet felhasználónak még sokáig problémát okoz, hogy egy keresett információhoz hogyan juthat hozzá, egyáltalán hol keresse. Elképzelhető például, hogy létezik egy file, ami az adott témáról szól, de hogyan találjuk ki, hogy milyen néven mentette el valamelyik távoli rendszeradminisztrátor? E nélkül az Archie szolgáltatás viszont (szinte) semmit sem ér. Jó lenne, ha magukban a file-okban kereshetnénk valamilyen kulcsszó szerint. Az ilyen, és hasonló problémák kezelését teszi lehetővé a Wais szolgáltatás.

A Wais rendszer két részből áll: a kliens, és a server. A felhasználó a kliens programmal áll kapcsolatban, amely segít a kérdés megfogalmazásában, és automatikusan lekérdezi a megfelelő server-eket. Egy server a saját diszkjein tárolt dokumentumokban tud keresni, ezért kell legyen valamilyen központi sever, amely magukról a Wais server-ekről szolgáltat információt. Ha a server-ekről szóló információt jól készítjük el, akkor ezekben a dokumentumokban ugyanúgy a Wais eszközeivel kereshetünk, mint bármely más dokumentumban.

Egy ilyen keresés tehát általában két (vagy több) lépésből áll. Először lekérdezzük a központi server-t, hogy a keresett témakör melyik másik server-en található, majd feltesszük ugyanezt a kérdést a válaszként kapott server-nek, illetve server-eknek. A válasz egy sorrendbe rakott dokumentum lista, az alapján, hogy a keresett kulcsszó melyikben fordul elő relatíve leggyakrabban. A listáról ez után kiválaszthatjuk a minket érdeklőket, és a Wais kliens automatikusan áthozza a kért dokumentumot, ha kell anonymous FTP-vel, vagy más egyéb módszerrel, és megjeleníti a képernyőnkön. Kellően gyors hálózat esetén a felhasználó szinte észre sem veszi, hogy a keresett információ esetleg a világ másik végéről érkezik.

Az itt ismertetett Internet szolgáltatások szükségszerűen csak töredékét képezik az Interneten ténylegesen elérhető szolgáltatásoknak. Rendszeresen megjelenő több száz oldalas – gyakran elektronikus formában terjesztett – kiadványok készülnek, hogy a felhasználókat tájékoztassák a lehetőségekről. Egy ilyen kiadvány az "Internet Resource Guide", amely Anonymous FTP-vel magyarországi szolgáltatógépekről is lekérhető. Egy másik fontos információforrás a NetNews, amelyben mindig az éppen aktuális, legfrissebb újdonságokról tájékozódhatunk.

A 90-es évek UNIX-a a Solaris 2 tükrében.
(Béta verzió)

Előadó: Erdélyi Ernő
ICON Kft

Összefoglalás:

A Sun legújabb UNIX alapú operációs rendszere a Solaris 2 egyszerre képviseli a folyamatosságot és a megújulást. Az általa nyújtott hármas szolgáltatás csoport: a SunOS 5.0, mely a System V Release 4-el kompatibilis UNIX, az ONC (Open Network Computing) hálózati csomag és az Open Windows grafikus felhasználói csatoló híven jellemzi, hogy mit várhatunk a 90-es évek UNIX rendszereitől. A tanulmány a Solaris 2-t alkotó egyes technológiákról ad korántsem teljes áttekintést. Az itt közölt tanulmány szöveg nem tekinthető véglegesnek, mivel a termék újdonsága miatt (1992 őszén jelenik meg a piacon) az előadás végleges tartalma a szöveg leadása után áll össze.

Bevezetés

A Sun 1982-ben alakult, és tiz év alatt páratlan karriert befutva bekerült az első 150 legnagyobb vállalat és az első 20 számítástechnikai cég közé. Ma első helyen áll a UNIX alapú munkaállomás/szerver piacon.

A Sun UNIX-változata a SunOS a BSD UNIX-ból származott (a Sun egyik alapítója Bill Joy egyben a BSD UNIX egyik atyja is), és ezt különösen a hálózati szolgáltatások és a grafikus képességek területén fejlesztette tovább. Az így kialakult komplex operációs rendszer az utóbbi időben a Solaris gyűjtőnevet kapta. A legújabb verzió, a Solaris 2.0 sokszempontból mérföldkő a Sun történetében: a termék jónéhány régebbi technológiát továbbfejleszt, és egyesíti azokat számos újjal. Az így létrejött robosztus, funkciógazdag és felhasználóbarát rendszer a 90-es évek egyik meghatározó platformját jelentheti. A Solaris külön érdekessége, hogy a Sun által használt SPARC rendszereken kivül, a széles körben elterjedt 386/486 alapú PC-re is megjelenik.

A Solaris három fő része a System V Release 4 (SVR4) kompatibilis SunOS 5.0, az ONC (Open Network Computing) hálózati rendszer és az Open Windows hálózati grafikus csatoló, melyek mindegyike számos érdekes technológiát tartalmaz. Ezek közül csemegézünk a

technológiát, amely a 90-es évek folyamán fontos szerephez jut a UNIX alapú rendszereknél.

Moduláris kernel

A UNIX születésének idején a következetesen végigvitt alapelvek miatt a rendszer és maga a kernel egyszerű és kis méretű volt. A fejlődés folyamán viszont rengeteg többlet funkció épült be, ami a kernel méretét is több megabyte-nyira növelte. Egy tipikus UNIX kernel rengeteg fizika eszköz kezelését tartalmazza, akár több hálózati architektúrát képes kezelni és számos egyébb funkciót támogat. E funkciók jelentős része egy konkrét installációnál kihasználatlanul marad, de mégis rezidensen foglalja a memóriát.

A probléma megoldása a moduláris kernel lehet, amely igény szerint tölti be az egyes funkciókat megvalósító modulokat. Az alap mudul, mely az ütemezést és a további modulok betöltését végzi mindenképpen elindul, ezen túl egy konfigurációs file-ban állítható, hogy mi az ami még egyből betöltödjön; a többi modul igény szerint indul.

Ez a fajta moduláris felépítés bővítés esetén feleslegessé teszi a hagyományosan használt kernel újrafordítást is. Egy új funkció megvalósításához csak egy új kernel modult kell beszerezni, amely ezután egyből müködőképes. A változtatható kernel paramétereket (maximális processz szám stb.) pedig egy konfigurációs file-ban adhatjuk meg.

Szimetrikus multiprocesszor rendszerek és multi-threading

Több évtizedes kutató munka után a multiprocesszoros számítógépek ma már egyre inkább a számítástechnika fő csapás irányába kerülnek. A multiprocesszoros rendszerek megvalósításának a megfelelő hardver mellett alapvető követelménye a hardvert hatékonyan kezelő szoftver. A fő kérdés az, hogy milyen finoman képes a rendszer a feladatokat az egyes processzorok között megosztani. Minél finomabb ugyanis a megosztás, annál egyenletesebb a proceszorok terhelése, és annál hatékonyabb a multiprocesszor működése. A megosztás lehet procesz, thread, vagy még nagyobb finomságú. A legegyszerűbb megoldás a legkevésbbé finom procesz szintű megosztás. Mivel egy tipikus UNIX rendszeren jónéhány procesz fut, ezek elosztása a proceszorok között sok feladatnál, ahol a számítási-adatfeldolgozási feladatok vannak

túlsúlyban hatékony megoldást kinál. A probléma csak az, hogy a UNIX-ban az I/O feladatok egyetlen proceszbe, a kernelbe vannak koncentrálva. Igy a procesz szintű megosztás az I/O intenziv feladatokat nem gyorsítja hatékonyan. Ilyenek tipikusan az adatbázis kezelések és a hálózati szerver funkciók. Ebben az esetben a kernel-t egy időben csak egy proceszor futtathatja. Emiatt az aszimetrikus kernel kezelés miatt az ilyen rendszereket aszimetrikus multiprocesszoroknak szokás nevezni.

A I/O intenziv feladatok gyorsítását a szimetrikus multiprocesszorok oldják meg. A megoldás szoftver alapja az, hogy a proceszek kisebb egységekre, un. thread-ekre bonthatók, melyek külön-külön is futhatnak így a több processzor között megoszthatók (természetesen ehhez a proceszeket is úgy kell megírni, hogy elkülöníthető thread-eket tartalmazzanak). Így például a kernelen belül külön thread lehet az ütemező (sheduler), a diszk kezelő, az ethernet kezelő és a többi device kezelő, és emiatt az egyik proceszoron futhat a sheduler, a másikon a diszk kezelő, a harmadikon az ethernet kezelő és a negyediken egy felhasználói program. Ez a szimetrikus megoldás már az I/O intenziv feladatokat is hatékonyan gyorsítja, de további gyorsítást adhat más jellegű programoknak is. Ugyanis nem csak a kernel lehet multi-threaded, hanem bármely procesz, és a multi-threaded proceszek thread-jei párhuzamosan futhatnak az egyes proceszorokon.

A Sun 1991-ben jelent meg először multiprocesszoros hardverrel, mely egyszerre támogat szimetrikus vagy aszimetrikus szoftver technikát. A korábbi operációs rendszer verzió, a Solaris 1 viszont csak az aszimetrikus multiprocesszor technikát támogatta. A Solaris 2 egyik fontos újdonsága a szimetrikus multipricesszor támogatás, valamint a multi-threaded programok fejlesztésének támogatása.

Real Time támogatás

A SunOS 5.0 tartalmazza a SVR4 összes valós idejű jellemzőit, valamint több más fontos valós idejű kiegészítést. Mindezek célja a determinisztikus válaszidő, mely a valós idejű alkalmazások alapvető kívánalma. A legfontosabb jellemzők a következők:

Fix prioritású valós idejű proceszek

Felhasználói procesz prioritás-manipulációja

Nagy felbontású timer-ek

Teljesen "preemtive" ütemezés

Procesz prioritás öröklése

Determinisztikus és garantált késleltetés

Szerver funkciók

Ahogy a UNIX rendszerek népszerüsége nő, egyre inkább betörnek a nagy ügyviteli rendszerek területére is, amelyeket hagyományosan a cégspecifikus rendszerek uraltak. Ezek az alkalmazások viszont új követelményeket vetnek fel a rendszerek, különösen a szerverek megbizhatóságával és rendelkezésre állásával kapcsolatban. Egy nagyvállalati vagy intézményi szerver esetén nem engedhető meg, hogy bármilyen okból (akár éjszaka is) hosszabb időre leálljon a működése.

Vannak továbbá olyan rendszerek, melyek egyáltalán nem türnek üzemszünetet, mint pl. egy bank, vagy egy tőzsde központi tranzakciós gépe. Az utóbbi kivánalmakat csak igen drága és tipikusan zárt rendszerek, az un. hibatürő rendszerek elégítik ki, mig az előbbi esetben, ahol hiba esetén egy-két perc kiesés még megengedhető jó megoldást kinál egy megfelelő elemekkkel kiegészített UNIX szerver is. Ez utóbbiak az un. nagy megbizhatóságú rendszerek.

A kiesés oka alapvetően háromféle lehet:

Környezeti hibák: melyekre a rendszer adminisztrátornak nincs ráhatása: pl. áram kimaradás, vagy a kommunikációs vonalak hibája (lásd Magyar Posta).

Rendszer hibák: hardver vagy szoftver problémák, ami miatt kiesik a rendszer: pl. diszk, diszk kontroller, CPU hiba.

Rendszer karbantartás: Egy tipikus UNIX rendszerben üzemszünetet kell elrendelni az archiválások, vagy diszkek átparticionálása esetén. Ez az az ok, amire az ember kevésbbé gondol.

A Sun nem akar belépni a hibatürő rendszerek területére, de a nagy megbizhatóságú rendszerek kiépítéséhez számos elemet kinál. Ezek a következők:

Online Disksuite: lehetővé teszi, a diszk tükrözést illetve azt, hogy több diszkből egy nagy egyesített un. metadiszket hozzunk létre, melynek nagysága a diszkek nagyságának összege. Így akár több tiz gigabyte-nyi egybefüggő metadiszk (file-system) hozható létre. A metadiszk nagysága akár menet közben is ujabb fizikai diszkkel növelhető.

Backup Copilot: lehetővé teszi, hogy egy tükrözött diszket akár üzem közben is elmentsünk

DualPort IPI Disks: lehetővé teszi, hogy IPI diszkek egyszerre két gépre is kapcsolódjanak. A termék tipikus használata, hogy egy szerver mellett ott áll egy másik tartalék szerver, és mind a kettő ugyanazokra a tükrözött diszkekre kapcsolódik. Disz hiba esetén a tükrözött változat tovább él, míg szerver hiba esetén a tartalék gép boot-ol és átveszi a szerver szerepét.

A felsorolt három termék külön termékként kapható, de az első kettő kernel részekkel is rendelkezik, ami a Solaris 2 esetén már default-ban rendelkezésre áll. Mind a tükrözés, mind a több fizikai diszken átnyúló logikai diszk megvalósításának alapja a Virtual File System (VFS) technika, mely már a hálózatok megjelenése óta rendelkezésre áll. A hálózati file rendszerek megjelenésével megbomlott a hagyományos Unix File System (UFS) monolit uralma. Ugyanakkor, a UNIX egységes file-hierarchiája miatt követelmény a különböző file-system tipusok egységes kezelése. A konkrét feladatra a Sun általános megoldást keresett, ami a valódi és különböző file-system-ek fölé vont egységes Virtual File System lett. Ez a megoldás később jól jött a PC-s formátumú floppy-k, valamint a CD-ROM kezelése során is. És mivel a VFS kezdettől fogva általános megoldásra törekedett, ezért nemcsak a különböző tipusú file-system-ek kezelésére képes, hanem több primér file-system És illetve Vagy kapcsolatára is.

Mit is jelent ez a homályos kijelentés? Két file-system Vagy kapcsolata azt jelenti, hogy ha a hozzájuk tartozó VFS-re akarunk írni, akkor egyszerre mind a két diszkre irunk, vagyis tükrözünk. Két file-system És kapcsolata pedig azt jelenti, hogy a két file-system-et egyesítjük, és a VFS mérete a két file-system méretének összegével lesz egyenlő. De a megoldás általánossága miatt nem csak ez a két alapeset lehetséges, hanem ezek bonyolítása és kombinációja is. Létrehozhatunk pl. kettőnél több példányos tükrözést is, vagy kettőnál több diszket is egyesíthetünk. És létrehozhatunk egy olyan VFS-t is, melyben egyszerre használjuk a méretnövelő egyesítést és a biztonságot növelő tükrözést. Mivel a file-system-ek eltakarják a fizikai diszk tipusát, ezért a VFS alatt lévő file-system-ek különböző vagy különböző tipusú kontroller-en is lóghatnak. Az előző esetben tükrözösnél nem csak a diszk, hanem a kontroller

hibája is kiszürhető. Az így létrejött virtuális file-system-ek a metadiszkek.

Transport Independent Open Network Computing (TI-ONC)

A UNIX világ legelterjedtebb hálózati rendszere a TCP/IP protokoll családra épülő ONC (Open Network Computing). A TCP/IP az OSI referencia model negyedik rétegéig a Transport rétegig definiálja a hálózatot. Az alap TCP/IP része továbbá néhány tipikusan WAN számára tervezett, de LAN esetén is használható alkalmazás (terminál emuláció, file transfer, mail stb.). A LAN-ok számára fontos szolgáltatásokat (pl. file sharing) viszont az alap TCP/IP nem valósítja meg. Erre a problémára kinált megoldást a Sun által a 80-as évek első felében kifejlesztett, és azóta széles körben elterjedt ONC (Open Network Computing) rendszere. Az ONC nem egyszerűen néhány hálózati szolgáltatás halmaza, hanem egy komplett hálózati architektúrát kinál, megvalósitva az OSI referencia model felső három szintjét. A TCP/IP-hez hasonlóan nem csak az volt a célja, hogy a UNIX-os rendszerek hálózati igényeit megoldja, hanem heterogán számítógépes hálózatok megvalósítására kínál megoldást.

Az OSI model Session szintjét az RPC (Remote Procedure Call) valósítja meg. Az RPC a nehezen, tipikusan socket-tel kezelhető Transzport réteget elrejti a programozó elől, és egy jól használható, a megszokott eljárás híváshoz hasonló felületet nyújt a processzek közti hálózati kommunikáció számára. Az RPC kliens-szerver modelt használ, melyben a kliens az eljárás hívó és a szerver az, aki az eljárást végrehajtja. Az RPC-t kezdettől fogva úgy tervezték, hogy ne legyen szigorúan a TCP/IP-hez kötve, alá más protokolok is betehetők legyenek. Ez a tulajdonsága a Solaris 2-ben teljesül ki igazán, mivel a hagyományos BSD socket-re épülő implementációt a Sun lecserélte a System V és X/OPEN kompatibilis TLI (Transport Level Interface)-re épülő implementációra. A TLI alapvető jellemzője, hogy program elől eltakarja a Transzport szint és így az ez alatti szintek konkrét megvalósítását, és így azt is lehetővé teszi, hogy az alkalmazás alatt megváltoztassuk a hálózat tipusát. Így az eredetileg TCP/IP-re implementált ONC áttehető az OSI hálózatokra, és így egy már jól ismert LAN megoldás-csomagot kinál az alkalmazásokban szükölködő OSI-nak. De ugyanígy implementálható az ONC más hálózatok pl. DECnet vagy Novell fölé is.

Az RPC további érdekes jellemzője, hogy a TCP/IP UNIX implementációjánál szokásos (rlogin, rsh, rcp), és nem túl erős biztonsági rendszer mellett egy ennél sokkal szigorúbb módszert is kínál. A hagyományos TCP/IP-s UNIX-os implementáció alapvető problémája, hogy "bemondásos" alapon működik: a szolgáltatást kérő kliens procesz közli a szerver processzel, hogy melyik gépen, melyik felhasználó neve alatt fut, és ezt a közlést a szerver procesz ellenőrzés nélkül elfogadja. Az RPC biztonságos változata a secure RPC viszont egy megbizható ellenőrzést alkalmaz, amely megakadályozza a szolgáltatásokhoz való jogosulatlan hozzáférést.

Az efölött, a Representation szinten elhelyezkedő XDR (External Data Representation) a különböző géptipusok eltérő ádatformátumait konvertálja egy egységes hálózati formátumra, így támogatva a heterogén hálózatok megvalósítását.

A fenti hálózati strukturára épülő legismertebb alkalmazás az NFS (Network File System), mely a file sharing funkciót valósítja meg. Az ONC további sztandard alkalmazásai az NFS mount-olást hivatkozás esetén automatikusan végrehajtó Automounter, a következő részben ismertetendő NIS+, a távoli parancsvégrehajtást megvalósító REX (Remote Execution) és a hálózati file illetve rekord lokkolást biztosító Lock Manager.

A Network Information System Plus (NIS+)

A TI-ONC-re épülő egyik fontos alkalmazás a NIS+ (Network Information System Plus, korábban ismert nevén Yellow Pages), a már jól ismert NIS továbbfejlesztése. Nagyobb hálózatokban tipikus probléma, hogy jónéhány olyan adatbázis létezik, melyeket minden host-on tárolni kell, de amelyek tartalma azonos. Ilyen adatbázis például a hálózati gépek, felhasználók és csoportok adatbázisa. Az azonos adatok több helyen való tárolása eleve ellentmondás, hiszen a számítógépes Murphy törvény szerint, ha ugyanazt az adatbázist két különböző helyen tároljuk, a két változat az idő múlásával egyre inkább el fog térni egymástól. Másrészt rengeteg plusz munkával jár, ha ugyanazt az adatbázist minden változtatás után rengeteg helyen kell módosítani. A probléma megoldása egy központilag karbantartott, de a hálózaton belül bárhol elérhető adatbázis. Ezt valósítja meg a NIS, melyet az NFS-hez hasonlóan, számos UNIX-os gyártó átvett a Sun-tól.

A NIS szempontjából háromféle géptipust különböztetünk meg: master server, slave server és client. A master server tárolja az adatbázisok eredeti példányait, és csak itt történhet ezek karbantartása. A slave server-ek csak az adatbázisok kódolt másodpéldányait tárolják, és tőlük az adatok szintén lekérdezhetők. A kliensek pedig egyáltalán nem tárolják az adatbázisokat, ezekhez csak a szervereken keresztül férnek hozzá.

Egy hálózaton belül egy master server és tetszőleges számú slave server illetve client lehet. A kliensek első lekérdezésük során hozzákapcsolódnak az egyik (a körkérdésükre leggyorsabban válaszoló) szerverhez, és mindaddig tőle nyerik az adatokat, amig az illető szerver egy kérésre a megadott időn belül nem válaszol. Ekkor a kliens újra egy körkérdést intéz a szerverekhez, és ezáltal új szervert keres magának. Látjuk tehát, hogy a több-szerveres megoldás értelme egyrészt a hibatürés megvalósítása (ha egy szerver kiesik, a kliensek egy másik szerverhez fordulhatnak), másrészt a szerverek leterheltségének csökkentése (nagy hálózatban, ha egy szerver leterheltség miatt nem válaszol időben, a kliensek új szervert keresnek).

A NIS+ több szempontból továbbfejleszti a NIS-t. A legfontosabb újdonság talán az alap TCP/IP-ből ismert Domain Name Services-hez hasonló hierarchikus domain struktúra. Eszerint egy nagy hálózaton belül hierarchikus csoportokat, domain-okat szervezhetünk, melyek azonos adatbázisokat használnak. Tehát pl. egy nagyvállalat vagy egyetem teljes hálózata a fő domain, és az egyes osztályok illetve tanszékek gépei egy-egy al domain-t alkotnak. Ezek után a fő domain-re közös lehet a host adatbázis (/etc/hosts), de más adatbázisok pl. a felhasználókra vonatkozó adatbázis (/etc/passwd) al domain-ről al domain-re különbözik. Ez a struktúra lehetővé teszi, hogy a nagy szervezetek kivánalmaihoz rugalmasan illeszkedő hálózati rendszer alakuljon ki. A NIS+ domain név struktúrája teljesen hasonló a Domain Name Services strukturájához, tehát egy hálózati fő domain neve pl. icon.hu, az al domain-ok neve pl. sale.icon.hu, marketing.icon.hu, support.icon.hu stb.

A NIS+ másik fontos újdonsága a megnövelt biztonság. A NIS+ a secure RPC-re épül, és így elődjénél sokkal nagyobb biztonságot nyújt. Az adatbázisokhoz való hozzáférés finoman, többszinten szabályozható. A hozzáférést megadhatjuk adatbázisonként (map),

soronként (entry) és oszloponként (column). Az adatbázisokat nem super-user is adminisztrálhatja.

A NIS+ felülről kompatibilis a NIS-sel: egy NIS+ szerver egyszerre képes NIS és NIS+ kliensek kiszolgálására, igy biztosítva a fokozatos átállást.

OPEN WINDOWS

A Solaris felhasználói csatolója az Open Windows. Ez tulajdonképpen gyűjtőnév, mely három szintet foglal magában: az X-Window/NeWS szervert, az OPEN LOOK grafikus felhasználói csatolót (Graphical User Interface, GUI) és a Deskset Tools eszközöket.

A mai UNIX rendszereknél megszokott módon az Open Windows is kliens-szerver architektúrát használ és hálózati működést tesz lehetővé. A kliens-szerver architektúrában a felhasználói program kliensek kommunikálnak a háttérben futó szerver processzel, amely kezeli az egyes grafikus device-okat: (display, mouse, keyboard). A szerver a device-ok okozta eseményeket (mouse mozgatás, gombnyomás és keyboard műveletek) közli a kliensekkel, mig a kliensek grafikus display műveleteket kérnek a szervertől. A hálózati működés pedig azt jelenti, hogy a szerver és a kliens proceszek akár különböző gépeken is futhatnak: más szóval lehetséges az, hogy egy gép előtt ülve, amin a szerver procesz fut olyan programokat indíthatunk és használhatunk amelyek egy másik, (akár más tipusú) gépen futnak.

A fő különbség az Open Windows szerver és a szokásos UNIX-os window szerverek között, hogy az előbbi nem egy egyszerű X-Window szerver, hanem egy kombinált X-Window/NeWS szerver. A NeWS a Sun saját fejlesztésű window szervere, melyet az MIT X-Window fejlesztésével nagyjából egyidőben alkotott meg, és annak konkurensének szánt. Mig az X-Windows szerver és a kliensek raszter szinten kommunikálnak, a NeWS szerver és a kliensek közös nyelve a jól ismert PostScript lapleíró nyelv. A PostScript nyelv sokféle előnyt biztosít: PostScript file-ok könnyű megjelenítését, könnyű skálázhatóságot, a nyomtató és a display egységes kezelését, stb. A UNIX világ mégis az X-Window rendszert fogadta el szabványként, és igy a Sun átvette azt és egyesítette a NeWS-zal. Az igy létrejött X-Window/NeWS (xnews) server egyszerre képes a szabványos X-Window és a NeWS kliensek kiszolgálására. Mivel az X-Window és a NeWS alapelvei számos ponton megegyeznek, ezért nincs szükség a szerver teljes megkettőzésére: az X-Window/NeWS szerver közös

ablak és esemény kezelőt használ, amihez kétféle interpreteren keresztül lehet kéréseket küldeni.

Ez a hibrid megoldás egyesíti a két rendszer előnyeit: az X-Window alá írt alkalmazások nagy számát és a NeWS-on keresztül a PostScript file-ok könnyű kezelhetőségét. Ez utóbbira példa a NeWSPrint; egy PostScript interpreter raster nyomtatók számára; a Pageview, mely lehetővé teszi PostScript programok editálását, display-en való megjelenítését és nyomtatását; valamint az Answer Book, mely a Sun rendszerek teljes dokumentációját nyújtja egy intelligens kereső rendszerrel kiegészítve PostScript formátumban, amely a képernyőn megjeleníthető illetve kinyomtatható.

A két interpretert két különböző socket páron keresztül lehet elérni (a socket egyfajta UNIX programozási struktúra processzek közti kommunikációra). Az interpreterekhez tartozó egyik (unix tipusú) socket az azonos gépen futó kliensek és a szerver gyors kommunikációját szolgálja, a másik (internet tipusú) socket a külön gépen futó kliensek és a szerver kommunikációjára használható.

Az Open Windows következő szintje az OPEN LOOK GUI, melyet a Sun az AT&T-vel közösen fejlesztett. Az OPEN LOOK egy jól definiált stilust határoz meg a felhasználói programok számára. Ehhez illeszkedik az OPEN LOOK window manager procesz (olwm). A window manager egy speciális kliens amely az ablakok és ikonok alapvető kezelését végzi (ikonba csukás, ablakká nyitás, mozgatás, ablakok méretének változtatása, előtérbe/háttérbe rakás, ujrarajzolás).

Az Open Windows harmadik rétege a Deskset Tools: 14 alapvető eszköz, mely lehetővé teszi, hogy az átlag felhasználó mindenféle UNIX ismeret nélkül használja a Solaris-t. A eszközök között találhatunk többek közt egy file manager-t, egy text editor-t egy multimédea mailtool-t, mellyel szöveget, hangot, grafikát és tetszőleges más file tipust küldhetünk el, egy tape tool-t egy print tool-t, egy kalkulátort, egy audio tool-t, mellyel hangot vehetünk fel, szerkeszthetünk, elmenthetünk és visszajátszhatunk.

Az Open Windows programozást segíti a külön megvehető Developer"s Guide CASE eszköz, melynek segítségével programozás nélkül, grafikus elemekből építkezve alkothatók meg a program

ablak egyes elemei (menük, scrollbar, kontroll mezők stb.), melyeket aztán a Developer"s Guide C nyelvű könyvtár hívásokra fordít le. Igy a programozó mentesül a felhasználói csatoló nehézkes programozásától és minden energiáját a program funkcióinak szentelheti.

A továbblépés irányai

A konszolidáción túl a legfontosabb két továbblépési irány a multimédia és az objektum orientált környezet. A Sun rendszerek már ma tartalmaznak mikrofont, hangszórót és egy 8 bites AD/DA chipet. A legújabb SPARCstation 10 pedig 16 bites CD minőségű AD/DA-t valamint ISDN interface-tartalmaz. További olcsó kiegészítő a VideoPix kártya és szoftver, mely lehetővé teszi video álló képek bevitelét. A feladat az ezekre épülő alkalmazások illetve szoftver fejlesztő eszközök megalkotása. Az első ilyen alkalmazások egyike az Open Windows-ban található Audio tool, mely hangfelvétel készítését, lejátszását, elmentését és egyszerű szerkesztését teszi lehetővé, valamint a Mail tool, amely a hagyományos karakteres szövegek mellett képes hangot, képet, grafikát, vagy tetszőleges dokumentum formátumot (pl. Lotus, Frame Maker stb) küldeni és fogadni.

Az objektum orientált környezet alapelemeit már most tartalmazza az Open Windows. Egy file feldolgozást kezdeményezhetjük a megfelelő tool ablakába, vagy ikonjába való mozgatással. Ha egy file-t pl a szemétkosárba mozgatjuk, akkor az eredeti helyéről törlödik (de a szemétkosárból visszaállítható), ha a Mail Tool-ba mozgatjuk, akkor a levél mellékleteként elküldjük, ha a Print Tool-ba, akkor nyomtatásra kerül, ha a Tape Tool-ba, akkor archiválásra kész. Ugyanigy egy file megnyitása is a file tipusának megfelelően történik: Texteditor-ban a szöveges file-ok, Audio Tool-ban a hang file-ok, a Pageview Postscript interpreter-ben a Postscript file-ok, a Snapshot képkezelőben a képek. Az objektum orientált környezet továbbfejlesztését a Sun a HP-vel közösen végzi, és a projekt célja a fejlesztés szabványként való elismertettése

Az UI - ATLASZ koncepció

Előadó: Ehry Attila

1. Bevezetés

A UNIX International 1988. decembere óta dolgozik a UNIX System V operációs rendszer fejlődési irányának meghatározásán. Ez a munka igen gyümölcsözőnek bizonyult, és sikerült áttekinthető fejlődési irányt szabni a UNIX System V-nek. Részletesen meghatározták a követendő technológiákat, és behatárolták ezek rendelkezésre állásának idejét. Mindezeken túl a UNIX System V a nyílt rendszerek világában a legelterjedtebben használt operációs rendszer, részesedése itt 80% felett van. Egyedül 1991-ben 1,1 millió gépet adtak el UNIX System V-el.

1991. elején a UNIX International tagsága felvetette, hogy terjessze ki a UNIX International a tevékenységi körét az operációs rendszeren túlra, és írjon elő teljes rendszerszoftver környezetet a nyílt rendszerek számára. Ennek a munkának az eredménye az UI-ATLASZ, melyet 1991. szeptemberében jelentettek be.

Az UI-ATLASZ egy összefoglaló nyílt rendszer, amely egy teljes számítástechnikai környezet, vagy egy vállalat részére nyújtott rendszerszolgáltatások számára szabványos csatoló felületet ír elő, és biztosítja ezen kívül a jövőbeli elosztott alkalmások fejlesztéséhez a megfelelő technológiákat. Az UI-ATLASZ ezt egy keret előíráson, amely megszabja hogyan kell a technológiákat a rendszerhez illeszteni, valamint szabványosított alkalmazói program csatoló-felületeken (API), protokollokon és adatformátumokon keresztül éri el.

Ez a szabványosítás két jelentős eredménnyel jár:

- A rendszerprogramozók tetszés szerint beépíthetik saját programjaikba a UNIX International és a UNIX System Laboratories által az API-t támogató referencia technológiaként rendelkezésre bocsátott szoftvereket, ezzel jelentősen csökkentve a fejlesztés időtartamát és ezzel együtt a termékeik piacra kerülési idejét, vagy készíthetnek olyan saját szoftvert, amely eleget tesz

az API szabványnak.

- Az alkalmazói programozók építhetnek az API-re, még mielőtt ez a tecnológia valójában rendelkezésre állna, ezzel termékeik piacra jutását jelentősen felgyorsítják.

A "UNIX International 1992 Roadmap for UNIX System V and UI-ATLAS" - a továbbiakban Roadmap az első olyan közlemény, amely a UNIX International kibővített célját és a UI-ATLASZ modell leírását ismerteti. Az említett közleményben szereplő szolgáltatások jó része különböző forrásokból máris rendelkezésre áll. Az ilyen esetekben a UNIX System Laboratories együtt fog működni a már létező szolgáltatások fejlesztőivel, hogy biztosítsa, hogy a referencia megvalósítás az UI-ATLASZ minden előírásának megfeleljen, hozzáférhető és rendszerbe illeszthető legyen.

A X/Open's Portability Guide 4 (XPG4) és más ide vonatkozó szabványnak való megfeleltetés biztosítja, hogy a UNIX System V és az UI-ATLASZ összeegyeztethető marad most és a jövőben is minden lényeges gyártó saját szabványával.

A UNIX International és a UNIX System Laboratories által alkalmazott módszer biztosítja, hogy az UI-ATLASZ modell minden összetevőjére referencia technológia álljon rendelkezésre.

Néhány, a Roadmap-ben leírt, és a UNIX System Laboratories által referencia technológiaként forgalmazott termék:

- Összefogó rendszer szervezés
- Új tranzakció feldolgozás
- Az OSF osztott számítástechnikai környezet (Distributed Computing Environments (DCE) támogatása
- A UNIX system V hálózatkezelés kiemelése (Sun ONC-je)
- UNIX System V "desktop" változat

A Roadmap folytatása és kiteljesítése a nyílt rendszerek közösségéhez tartozó különböző szervezetek együttműködésének. A Raodmap, mint keret elfogadásával ezek a szervezetek együttműködhetnek, miközben önállóan fejlesztenek.

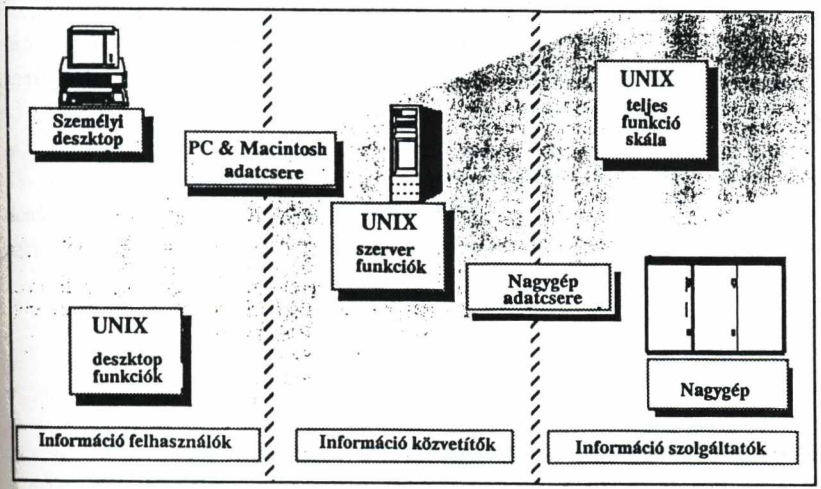
2. Számítástechnikai környezet és a UNIX rendszer

Az 1991 Roadmap - irányelveiben a UNIX International azt mutatta be, hogy az ismertetett különböző lehetőségek hogyan használhatók deszktop, osztott és nagygépi környezetben. Az 1992 Radmap irányelvei azt taglalják, hogy a fenti környezetek hogyan kapcsolódnak egymáshoz egy vállalati rendszerben.

Egy vállalatnál a számítógépek alapfunkcióit használják. Ezek a funkciók a következőkben foglalhatók össze:

- Információ felhasználók - segíti a felhasználót a munkájában, betekintést nyújt a vállalati szintű információkba.
- Információ közvetítők - kapcsolat teremtés az információ felhasználók és az információ szolgáltatók között, az általánosan használt adatok tárolását látja el.
- Információ szolgáltatók - védett, nagy megbízhatóságú tranzakciók és adatfeldolgozás.

Az 1. ábra azt mutatja be, hogyan szolgáltatja a UNIX System V a fenti funkciókat egy vállalatnál, amely gépparkja zárt és nyílt rendszerű számítógépeket egyaránt tartalmaz.



1. ábra
Vállalati osztott feldolgozás

Ezeket a funkciókat akár milliárd, osztott feldolgozási környezetben működő rendszer, vagy csak egyetlen gép is nyújthatja. Függetlenül a konfigurációtól, ez a három funkció a legtöbb installáció esetében érvényes.

A UNIX International csoport tagjai tiszteletben tartják vállalt kötelezettségüket, támogatják az 1991 Roadmap-ban előírt környezetek megvalósítását, összpontosítják erőfeszítéseiket ezen környezetek előírásainak betartására, ezzel lehetővé téve a különböző információs technológiák zökkenőmentes összekapcsolását, mint ezt a fenti vállalati modell is mutatja.

A UNIX Smegvalósításához megkívánt technológia a következőkben foglalható össze:

- Információ felhasználók - a UNIX System V deszktop funkciók könnyen használható tulajdonságokat, és a személyi munkát elősegítő eszközöket nyújtanak egy olyan konfigurálható szoftver környezetben, amely a vállalati információ zökkenőmentes áramlására van tervezve.

- Információ közvetítők - a UNIX System V szerver funkciók lehetővé teszik az erőforrások és alkalmazások átlátszó, zökkenőmentes megosztását heterogén rendszerekben, ily módon információt közvetítenek teljes vállalati szinten.

- Információ szolgáltatók - a UNIX System V teljes számítástechnikai funkcióskálája egy nagy teljesítményű, osztott on-line tartanzakció feldolgozó (OLTP) környezet minden szolgáltatását nyújtják, információkat szolgáltatnak a vállalat egésze számára.

A fenti követelmények kielégítése céljából a UNIX International tagok egy sor célt tűztek ki a UNIX System V elé. Az 1992 Roadmap előirányzatai ezen célok elérésének lehető legrövidebb útját és idejét vetítik elénk. A továbbiakban ezen célok rövid leírása következik.

2.1 UNIX deszktop funkciók

A deszktop rendszerek a számítástechnikai ipar legerőteljesebben fejlődő ágazatát jelentik az 1990-es években. A 80-as években a deszktop rendszerek nagy része a személyi operációs rendszerekre, mint az MS-DOS vagy Macinthos alapult, a 90-es évek során azonban egyre több ilyen rendszer épül UNIX System V-re.

Hagyományosan a deszktop rendszerek többnyire a személyi munkát segítő eszközökkel voltak ellátva, mint például szövegszerkesztők, táblázatkezelők. A deszktop rendszereknek ezek a hagyományos feladatai a 90-es években jelentősen megváltoznak annak a felismerésnek hatására, hogy egy szervezeten belül az információ kulcs jelentőségű, és hogy a számítástechnika vállalat szintű alkalmazása nagyobb rugalmasságot és kapacitást igényel a deszktop rendszerektől is. A nagyobb rugalmassággal és kapacitással együtt előtérbe került a deszktop rendszerek beillesztése a vállalati rendszerbe. Ezek a kívánalmak kiemelik az együttműködés, a kapcsolódás és az osztott feldolgozás jelentőségét, lehetővé teszik, hogy a deszktop rendszer egy termelékeny, önálló munkaállomás legyen, érhesse el a vállalati szintű információkat és földrajzilag szétszórt más deszktop rendszerekhez kapcsolódhasson. Feltehetően a legfontosabb, hogy a deszktop rendszer könnyen használható legyen. Az egyszerű, könnyen használhatóságot a 90-es években képek, és nem szavak fogják biztosítani.

Az alapszoftver technológia, amelyet ez az új generációs számítástechnika igényel, multimédia technikákat fog használni és osztott, tárgyorientált környezetre épül. A UNIX System V már jelenleg is ilyen kiforrott környezetet biztosít, amelyre már fejlesztik a tárgy-orientált alkalmazásokat. Mindezeken túlmenően a UNIX System V-nek nincs párja a kapcsolhatóság és hálózati szolgáltatások terén akár személyi, akár más nyílt rendszerek irányában. Mindezek figyelembevételével alakították ki a UNIX International tagok a deszktop funkciókkal szembeni követelményrendszerüket.

Az 1992 Radmap előírásai megkívánják, hogy a UNIX System V szolgáltasson olyan kulcs deszktop funkciókat, mint a tárgyorientált deszktop szolgáltatások, virtuális függetlenség, osztott rendszer szervezési alkalmazások.

2.2 UNIX szerver funkciók

A UNIX System V hagyományosan igen hatékony szolgáltatásokat nyújt a vállalat különböző részeinek kiszolgálásában. Rugalmas felépítése és beépített multiprogramozási lehetőségei kedveltté teszik a hálózati alkalmazások fejlesztői körében. A 90-es évek alkalmazásai még szélesebb körű hozzáférést igényelnek a szolgáltatásokhoz és információkhoz a vállalaton belül, a hangsúly pedig tárgy-orientált programozási módszerek használatával ilyen jellegű alkalmazások fejlesztése felé tolódik. A UNIX International tagok a fentiek figyelembevételével tűzték ki a szerver funkcióval szemben támasztott követelményeket.

Az 1992 Roadmap ezen célok elérésére érdekében olyan jellemvonásokat ír elő, amelyek kidomborítják a UNIX System V azon képességét, hogy együtt tud működni mind személyi, mind nyílt rendszeri elemekkel. Személyi rendszerek számára az 1992 Roadmap előírja, hogy a UNIX System V úgy kapcslódjon hozzájuk, érje el és kérdezze le adataikat, hogy ne legyen szükség a személyi rendszer módosítására.

Nyílt rendszeri környezetek esetén az előírás a UNIX System V számára az, hogy támogassa az osztott, tárgy-orientált alkalmazásokat egy sor rendszer szolgáltatáson keresztül anélkül, hogy bármit változtatni kellene a már üzemelő alaprendszeren. Ezen felül elő vannak írva olyan szolgáltatások is, mint a központilag adminisztrált osztott rendszer irányítás és adatbiztonság.

2.3 UNIX teljes számítástechnikai funkció skálája

A tranzakciók teszik ki a legtöbb üzleti cég fő tevékenységét, ezek megfelelő szervezése jelentős versenyelőnyöket jelenthet. A UNIX System V teljes funkció skálája már most is jelentősen támogatja a tranzakció feldolgozást. A UNIX International tagok célja, hogy a teljes funkcióskálát jelentősen kibővítik a tranzakció feldolgozás irányába.

Az 1992 Roadmap előírja, hogy a UNIX System V ugyanolyan jelentős kereskedelmi szolgáltatásokat nyújtson, mint ami előtte csak nagy rendszerekre volt jellemző. Olyan funkciókat foglal magában, mint a tranzakció szervezés, rendszer szervezés, hibakezelés, szimmetrikus multiprogramozás, kapacitás tervezés és biztonság. A fő erőfeszítések az on-line tranzakció feldolgozás irányába hatnak, de más feldolgozási fajtáknak mint döntés támogatás és kötegelt feldolgozásnak is jelentős szerep jut. Jóllehet önálló rendszerek is tagjai ennek a környezetnek, a hangsúly az osztott OLTP-n van, ahol a rendszer szolgáltatások számítógép hálózatban szétszórva jelennek meg, és egy osztott tranzakció felügyelő egy egyesített tranzakció rendszerként szervezi őket. Megjegyzendő, hogy a UNIX System V már ma is jelentős részét nyújtja az 1992 Roadmap-ban előírt szolgáltatásoknak.

3. A UNIX International UI-ATLAS, a nyílt döntési folyamat

A folyamat, amely segítségével a UNIX International tagok megszabják a UNIX System V fejlődési irányát a "nyílt döntési folyamat" elnevezést kapta. Ennek a

folyamatnak a lépései a következők:

- A Roadmap albizottság (RMSC) olyan dokumentumok, mint az X/Open, Open Systems Direktive és más ipari követelmények kiértékelésével vizsgálja a piac igényeit.

- Az RMSC szakértői bizottság, az érdekelt UNIX International tagvállalataiból verbuválódik, részletes piaci tanulmányokat készít meghatározott technológia területekről. Az RMSC kiértékeli az ezektől a bizottságoktól nyert információkat és kidolgoz egy prioritásokkal ellátott szolgáltatás listát, amit majd be kell iktatni a UNIX International Roadmap előírások közé.

- A UNIX International vezetői bizottság áttekinti ezt a listát, és a UNIX International végrehajtó bizottsággal közösen évente engedélyez egy módosított Roadmap előírást. A vezetői bizottság az érdekelt UNIX International tagok és más cégek képviselőiből munkacsoportokat állít össze, hogy a Roadmap előírások egyes elemeinek pontos leírását elkészítsék. Ezek a munkacsoportok általában 30-50 személyt számlálnak.

- Az egyszer ily módon engedélyezett termékekkel szembeni követelményeket a UNIX System Laboratories felé továbbítják felülvizsgálat céljából.

- A UNIX System Laboratories ezen követelményeknek eleget tevő termékleírást és részletes specifikációt (beleértve az API dokumentációkat is) készít.

- A termék specifikációkat a UNIX International munkacsoport ellenőrzi, hogy biztosítják-e az eredeti termék követelmények megfelelő megvalósítását.

- A UNIX International egy, a termékhez kapcsolódó API definíciót dolgoz ki, amely az ipar számára elérhető, alkalmazhatják az alkalmazások fejlesztésénél, beépíthetik az ipari szabványaikba.

- A UNIX System Laboratories a UNIX International tagokat a fejlesztési folyamat függvényében, megfelelő időközönként elsőként tájékoztatja az elérhető szoftverekről, amelyek megfelelnek az API-knek, eleget tesznek az előírásoknak és specifikációknak.

- A UNIX System Laboratories a technológiának egy referencia megvalósítását biztosítja.

4. Összefoglaló

Az 1992 Roadmap előírásait követő megoldások a teljes vállalat információ feldolgozási igényét kielégítik. A Roadmap-ben leírt összes szolgáltatás kifejlesztése és rendszerbe illesztése valószínűleg hosszabb időt igényel majd, de a UNIX International nyílt döntési folyamata biztosítja az ipar számára, hogy a specifikált összes funkció rövid időn belül hozzáférhető lesz különböző forrásokból, és hogy az összes megvalósítás meg fog egyezni az API-kkel és az előírt protokolokkal.

Azok a szervezetek, amelyek az UI-ATLASZ-nak eleget tevő terméket használnak, biztosak lehetnek afelől, hogy cégük számítástechnikai rendszerében történő bármilyen változás vagy bővítés, futó rendszerüket nem érinti, rendelkeznek majd a nyílt rendszeri környezet kulcs követelményeivel, mint az együttműködés, kompatibilitás, hatékonyság és szabad választás. A UNIX International elkötelezte magát a nyílt rendszereknek. A Roadmap előírások fejlesztési folyamatával ezt az elkötelezettséget nyilvánítják ki a UNIX International és tagjai az ipar irányába.

II.szekció: **ÚJ TECHNIKÁK ALKALMAZÁSA és OKTATÁSA**

MAGIC
Objektum-Orientált, Kódnélküli Alkalmazás Fejlesztő Rendszer UNIX és Kliens/Szerver környezetben

Szluha Márton és Tölgyesi István
ONYX Szoftverház Kft.

Ideális alkalmazás fejlesztő eszköz UNIX fejlesztők és felhasználók számára.

A MAGIC egyesíti azt a két elvet, hogy az alkalmazások elkészítésére és bevezetésére fordított idő illetve költségek minimalizálása mellett teljes nyitottságot érjünk el (operációs rendszer és platform függetlenség). Ez az a két fő elv, amely meghatározza a MAGIC jelenlegi és jövőbeni fejlesztési irányát.

Egy egyedülállóan automatizált programozási módszer, mely kifejezetten ideális a UNIX fejlesztők részére, mivel biztosítja az elkészült alkalmazás áthelyezhetőségét, a kereszt-platform alkalmazhatóságát, kliens/szerver architektúra alatt különböző adatbázis szerverek felhasználásával. A MAGIC már rendelkezésre áll a legtöbb UNIX platformon, operációs rendszer alatt és különböző DBMS kezelők alkalmazhatók a rendszerfejlesztésnél. A kliens/szerver szolgáltatás és a különböző adatbázis szerverek használata egyszerű módon lehetővé teszi VAX/VMS és DOS rendszerekhez való kapcsolódást.

Hatékonyság és Teljesítmény

Sok adatbáziskezelőről és alkalmazásgenerátorról állítják azt, hogy szükségtelenné teszi a programozást, azonban lényegében mégis kódgenerátorokról van szó. Habár ezzel a fejlesztő időt takarít meg a kódolás során, gyakran érzi úgy, hogy szükséges a generált program kód továbbfejlesztése. Az MSE cég MAGIC rendszere egy komplett, testreszabott fejlesztő környezet a professzionális fejlesztők számára, amely az új alkalmazások kifejlesztéséhez a legújabb fejlesztési technikákat nyújtó eszközöket - *relációs adatbázis model, gördülő (scrolling) és háttér (backround) ablakok, legördülő és felvillanó menük, objektumok, események kezelése* - biztosít. A MAGIC rendszert az emeli ki ezek közül a rendszerek közül, hogy ez a rendszer teljesen kiküszöböli a hagyományos procedurális programozási nyelv használatát.

A MAGIC nem hagyományos értelemben vett alkalmazás generátor, mivel nem kell kódot generálni, karbantartani vagy hangolni. Mindazonáltal, a MAGIC alkalmas arra, hogy ugyanolyan bonyolultságú rendszereket fejlesszenek ki és futtassanak vele, mint a hagyományos adatbáziskezelő rendszerekkel, anélkül, hogy a funkciók tekintetében bárhol meg kellene alkudni.

Hogyan tudja a MAGIC megsokszorozni a teljesítményt és ugyanakkor nagyobb hatékonyságot biztosítani mint egy hagyományos rendszer, anélkül, hogy procedurális programozási technikákhoz folyamodna? Tartalmaz egy adatbázis motort, mint ahogy egy szakértői rendszer tartalmaz egy következtetés generátort. A fejlesztő által megadott szabályok halmazához az adatbázis motor olyan mindennapi eljárásokat szolgáltat az adatbázis kezelési feladatokra, mint pl. a fájlok megnyitása és lezárása, rekordok olvasása és írása, hiba esetén az előző állapot visszaállítása, rendezések, rekordok visszakeresése és indexek létrehozása. Habár a procedurális programozás elmarad, a fejlesztőnek el kell végeznie a rendszer elemzését, az adatbázis struktúrájának megtervezését, és értelmeznie kell a rendszer különböző elemeinek működési sorrendjét és kapcsolatát. A MAGIC valamilyen adatbázis kezelőre (pl. C-Tree, C-ISAM, Ingres, Sybase) építve működik az összes jelentősebb UNIX környezet alatt (Pl. SunSPARC, DG AViiON, IBM RS6000, ULTRIX, SCO UNIX) , akár a kliens/szerver architektura felhasználásával.

Teljes DOS-UNIX-VMS áthelyezhetőség

A MAGIC egyedülálló támogatást nyújt a rendszerintegrátorok számára. Lehetővé teszi, hogy a UNIX, PC és a VAX felhasználó ugyanazt az alkalmazást és adatokat használja, ugyan azzal a felhasználói felülettel. A különböző rendszerek használata rendkívül kevés betanulást igényel a felhasználóktól, hiszen ugyanaz a Magic képernyő jelenik meg minden platformon. A rekordok foglaltságának kezelése teljesen automatikus, függetlenül attól, hogy az igény egy lokális UNIX-tól, egy UNIX állomostól, egy PC-től vagy egy VAX-tól származik. Ha egy PC felhasználó megváltoztat egy rekordot, akkor a változás valamennyi UNIX felhasználó képernyőjén azonnal megjelenik.

A MAGIC UNIX verziója teljesen compatibilis a többi MAGIC verzióval. A DOS-ban kifejlesztett rendszerek könnyen adaptálhatók és áthelyezhetők UNIX rendszer alá, a legtöbb alkalmazás minden módosítás nélkül fut a DOS vagy UNIX alatt.

A kliens/szerver szolgáltatás és a PC-UNIX kapcsolat biztosítja a felhasználónak a UNIX adatbázis használatát.

PC, VAX és UNIX felhasználók bárhol is helyezkednek el egy egyesített hálózaton képesek megosztani a UNIX adatbázisban lévő adatokat. Ezeknek a szétszórt adatoknak az összekapcsolása a MAGIC interaktív, nagy teljesítményű fejlesztő eszközével biztosítja az adatok feldolgozásának optimalizálását, a megnövelt adatbiztonságot és a nagyméretű UNIX adatbázisok és hardver platformok összefogását.

A MAGIC futtatása a UNIX-on szerver üzemmódban egy valós megosztott adatbázis szerver mechanizmust biztosít. Képes PC-ktől, kihelyezett UNIX számítógépektől vagy az adott UNIX gépen dolgozó felhasználóktól érkezett igények kiszolgálására.

Egy kliens/szerver architektúrában valamennyi MAGIC alkalmazónak azonos felhasználói felület áll rendelkezésre, és teljesen közömbös, hogy az adatok a saját számítógépén, vagy egy távol lévő számítógépen találhatók. Hasonlóan egy PC alkalmazó használni képes a UNIX gépet anélkül, hogy elhagyná a DOS-t vagy terminálemulátor programot használna. Egy UNIX-on kifejlesztett alkalmazás közvetlenül használható egy PC-n vagy egy helyi hálózat munkaállomásán.

Lehetőség van arra is, hogy egy alkalmazást DOS-ban futtassunk úgy, hogy az adatok és a programok a UNIX gépen helyezkednek el. Ezek a UNIX számítógépen lévő adatok megoszthatók más PC és UNIX alkalmazókkal, akiknek meg van a lehetősége, hogy ezekből az állományokból adatokat olvassanak vagy írjanak.

Kompatibilitás a legtöbb UNIX platformmal

A MAGIC lehetővé teszi a fejlesztést eltérő UNIX platformokon eltérő - egyedi vagy az ipari szabványnak megfelelő - UNIX verziókon keresztül. Valójában a MAGIC kompatibilis az összes fontosabb UNIX hardverrel, operációs rendszerrel és szabványos UNIX termékekkel, beleértve az IBM RISC/6000-t, Sun SPARC Workstation-t, 386/486 alapú PC-ket SCO UNIX rendszerrel, Digital DECstation-t és DECseries (Ultrix) gépeket, Data General AViiON sorozatot, Hewlett Packard 9000: 700/800 sorozatokat és az NCR 3000-t. További UNIX platformok és operációs rendszerek is integrálásra kerülnek.

A MAGIC adatbázis kezelő függetlensége maximális kihasználását biztosítja a UNIX DBMS rendszereknek.

A különböző gyártók által készített adatbázis kezelők front-end eszközeként a MAGIC egy teljesen új dimenzióját adja az adatfeldolgozó rendszerek fejlesztéseknek. A MAGIC teljesítő képessége lehetővé teszi nagy méretű alkalmazások felépítését, függetlenül az adatok méretétől, bonyolultságától és megosztottságától. A MAGIC irányítja a teljes programozási logikát, kezeli a felhasználói felületeket, a menüket és az adatokat. Lehetővé teszi, hogy a DBMS előnyös tulajdonságait kihasználjuk, megtartva az adatbiztonságot és adatvédelmet, a futtatható programok magas kiszolgálását és az adatbázis kezelő "hangolását" az erőforrások maximális kihasználásához és optimális elhelyezéséhez.

A MAGIC fejlesztők és felhasználók egyszerre érhetik el a különböző adatbázis kezelők által tárolt adatokat mind a helyi, mind a távoli számítógépeken. A MAGIC adatszótára biztosítja, hogy az alkalmazott adatbázis terminológiája szerint adjuk meg az adatok és a tárolás típusát, lehetővé téve a meglévő adatbázisokkal és alkalmazásokkal való kommunikációt. Az alkalmazók szabadon választhatnak a DBMS-ek és a tárolási típusok között.

A MAGIC-kel szállított fájlkezelő a Faircom cég terméke, amely c-tree technológiára alapozott, UNIX környezetben futó, ISAM típusú, nagysebességű és hatékony eszköz. A MAGIC biztosítja az adatbázis kezelést nem csak a c-tree, hanem a C-ISAM(Informix), az Ingres, az Oracle, a Sybase vagy más ismert DBMS rendszerrel egy Gateway segítségével.

Az MSE egy új szabványt teremtett a professzionális fejlesztők számára.

Kezdetben a MAGIC mint egy a negyedik generációs nyelveknél hatékonyabb eszköz vált ismertté a fejlesztők között. Mára az alkalmazások programozásának legkülönbözőbb területein megtaláljuk a MAGIC-et, a könyveléstől a pénzügyi rendszerekig, a közigazgatásban, a bankoknál és a gyártó cégeknél is. A MAGIC alkalmazók professzionális programozókból, rendszer fejlesztőkből és tanácsadókból, valamint VAR fejlesztőkből, szoftver házakból, tanácsadó cégekből és rendszerintegrátorokból áll. Az alkalmazók száma meghaladja a 100,000-ret, olyan cégeket beleértve, mint a Texas Instruments, a Union Carbide, a Wells Fargo Bank, az ENSZ, a Nestlé, a Hasbro, a Dunkin' Donuts, a Hewlett-Packard (UK), a British Sky Broadcasting és a Bank of Japan.

EGY X ALAPU 3D TERVEZÖRENDSZER

Padi Ferenc
Miskolci Egyetem Számítóközpont

Bevezetés

A tervezörendszert elsösorban oktatási célokra készítettem. Segítségével interaktív módon definiálhatók, transzformálhatók és kirajzolhatók egyszerübb háromdimenziós testek. A testek elemi testekböl (primitívekböl) építhetök fel halmazmüveletek segítségével. A rendszer arra is alkalmas, hogy szemléltesse ezen müveletek algoritmusát, különbözö fázisait. Igy a testmodellezés oktatására is felhasználható. A rendszert UNIX (Open Desktop Release 1.1) alatt fejlesztettem C nyelven, igy nagymértékben hordozható. Föbb építöelemei a következök:

- testmodellezö rendszer: GWB (Geometric Workbench)

- adatbáziskezelö rendszer: XDMS hálós adatbáziskezelö

- grafika: X Window

- user interface: OSF/Motif

A testmodellezö rendszer

A testmodellezö rendszer a GWB (Geometric Workbench) nevü modell egy implementációja , melyet Martti Mäntylä publikált "An Introduction to Solid Modelling" [1] címü müvében. Határfelület-modell , azon belül poliéder-modell. A poliéder lapjait , éleit és csúcsait C nyelvi struktúrák ábrázolják, a köztük levö kapcsolatokat pedig pointerek. Jellegzetessége, hogy mindegyik él két ún. félélböl áll: az élhez tartozó mindkét laphoz tartozik egy félél úgy , hogy irányítása megfelel a lap körüljárásának. (Innen az adatstruktúra elnevezése : Halfedge Data Structure.)

A test (solid) lapokból (face) áll. A lap határgörbéje (loop) egy síkbeli poligon, mely féléleknek (halfedge) egy ciklusa. Megengedett, hogy a lapban lyukak legyenek , igy egy laphoz tartozik egy külsö és n darab belsö határgörbe. A szomszédos lapokhoz tartozó két félélt egy él (edge) kapcsolja össze. A félélhez az a csúcs (vertex) van rendelve, amelyböl kiindul. A csúcshoz egy pont (point) tartozik, vagyis az x, y és z koordináták. Ez az egyetlen geometriai adat. Az összes többi a topológiai kapcsolatok leirására szolgál. Sok a redundancia, de ez megkönnyiti a hozzáférést. Például az egy laphoz vagy egy csúcshoz tartozó élek ciklusa könnyen bejárható. Ugyanigy könnyen elérhetö az egy élhez tartozó két szomszédos lap, stb.

Az adatstruktúra ilyen formájában biztosítja, hogy topológiailag helyes, zárt, irányítható poliéderfelületeket ábrázoljunk. Az adatstruktúra manipulációja legalacsonyabb szinten az ún. Euler-operátorok segítségével történik. Ezek egy-egy csúcs, él vagy lap beszúrását illetve törlését teszik lehetővé és automatikusan megőrzik a topológiai konzisztenciát. A geometriai helyességet a testgeneráló algoritmusoknak kell biztosítani. Az Euler-operátorokkal gyorsan generálhatunk elemi testeket. Ezek többnyire ún. "sweeping" primitívek , melyek egy síkbeli zárt görbe elmozgatásával keletkeznek. Az 1. ábrán egy eltolt és egy elforgatott primitívet láthatunk, ugyanabból a síkbeli görbéből származtatva. Forgástestek hasonló módon generálhatók. A sweeping fogalmába belefér a téglatest, gömb, henger, kúp, stb. vagyis azok a testek, melyek a CSG modellekben általában primitívekként szerepelnek.

A generált primitíveket különböző módokon kombinálhatjuk, hogy valóságos testeket nyerjünk. A legegyszerübb módszer a ragasztás. Ennél a testek két egybevágó lapját ragasztjuk össze úgy, hogy azok eltünnek. Ennél bonyolultabb müvelet a síkkal való metszés. Ez a sík mentén két különálló részre vágja szét a testet. Végül a legáltalánosabb müveletek a halmazmüveletek: az unió-, a metszet- és a különbségképzés. Az unióképzés például az áthatásszámításnak felel meg.

A GWB modell a test belsö, memóriában történö ábrázolására szolgál. Belöle egyszerüen számolhatók a test ún. volumetrikus jellemzöi (térfogat, felszín, stb.), valamint különféle rajzok (takartvonalas, takartfelületes, stb.) generálhatók. Azonban nem alkalmas arra, hogy a felhasználó interaktív módon, néhány paraméter megadásával definiálhasson testeket (pl. a henger definiálásához mindössze két paraméterre van szükség). Ehhez egy ún. CSG (Constructive Solid Geometry) modellre van szükség, mely lényegében egy bináris fa. Levelei a primitívek (az összes szükséges paraméterrel), csomópontjai pedig két ágat kapcsolnak össze egy halmazmüvelettel. Ez a modell a felhasználó számára is áttekinthetö és alkalmas arra is, hogy adatbázisban tároljuk.

Az adatbázis

Adatbáziskezelöként az XDMS nevü hálós adatbáziskezelöt (CODASYL szabvány, SZTAKI fejlesztés, lásd [2]) használtam. Ez különösen alkalmas geometriai struktúrák ábrázolására. Az adatbázis a rendszer felépítésének megfelelöen két részből áll: 2D és 3D. A 2D adatbázisban tárolja a rendszer a bevitt rajzi elemeket, a 3D adatbázisban pedig a test CSG-modelljét. Egy-egy testnek (alkatrésznek) egy-egy adatbázis felel meg.

Grafika

Grafikára egyrészt a 2D rajzok editálásakor, másrészt a 3D modellböl generált rajzok megjelenítésekor van szükség. Mivel nem állt rendelkezésemre semmilyen X alapú grafikus szabvány (pl. GKS), egyelöre az X Library pixelszintü grafikáját használtam. A szükséges magasabb szintü funkciókat

(koordinátatranszformációk, grafikus azonosítás, stb.) magam valósítottam meg.

User interface

A user interface (menük, dialógusok) kialakításához az X Toolkit-et ill. az OSF/Motif widget-készletet használtam fel. Hogy a rendszer minél interaktívabb legyen, a felhasználónak többnyire lehetősége van grafikus inputra szöveges helyett. (Pl. zoomoláskor kényelmesebb egy ablakot kijelölni a képernyön, mint azt koordinátákkal megadni. Hasonlóan egy primitív pozicionálásakor kényelmetlen lenne mindig koordinátákat begépelni, ehelyett kényelmesebb azt egy már meglevö primitívhez illeszteni.) A grafikus input jelenthet egy kurzor pozíciót (lokátor input), de jelenthet egy a képernyön azonosítható elemet is. A grafikus azonosítás markerek segítségével történik, melyek a rajzi elem középpontjában jelennek meg. Ezenkívül arra is törekedtem, hogy a felhasználó rögtön lássa, amit definiált vagy módosított (what you see is what you get). Természetesen nem lát azonnal árnyalt, takartfelületes képet a tárgyról, hiszen annak generálása idöigényes. Ehelyett egy új primitív definiálásakor annak csupán drótvázas rajzát látja. Végleges rajzot akkor kérhet, ha készen van a tárgy megtervezésével. Ekkor automatikusan megtörténik a halmazmüveletek elvégzése és a kért rajzok generálása.

A rendszer felépítése

A rendszer felépítése legegyszerübben a menüstruktúrán keresztül szemléltethetö. Két alrendszerre bontható. A 2D alrendszer segítségével a felhasználó elöállíthatja azokat a síkbeli görbéket, melyekre a testek definiálásakor szüksége van. Ez a 3D alrendszerben történik. Mindkét alrendszerhez külön X ablak tartozik. Fömenüiket az alábbiakban ismertetem.

A 2D alrendszer

A konstruktív modell lényege, hogy a felhasználó néhány paraméter segítségével definiálhasson primitíveket és azokat halmazmüveletekkel kombinálva kapjon bonyolultabb, valóságos testeket. Az primitívek egy síkbeli zárt görbéből származtathatók eltolással vagy elforgatással. A 2D alrendszer célja ezen görbék (kontúrok) elöállítása, melyhez a szokásos rajzeditori szolgáltatásokat nyújtja. A képernyön az elkészült kontúrok kurzorral azonosíthatók. A 2D alrendszer fömenüje a következö:

- **File**: 2D adatbázis kiválasztása, kilépés
- **Create**: rajzi elemek (szakasz, kör, stb.) kreálása
- **Delete**: rajzi elemek törlése
- **Transform**: rajzi elemek 2D transzformációja (eltolás, forgatás, tükrözés, nagyítás, duplikálás)
- **Zoom**: zoomolás

A 3D alrendszer

A 3D alrendszerben tervezheti meg a felhasználó a térbeli tárgyat (alkatrészt). Ez primitívenként történik. A primitívek többnyire egy síkbeli zárt görbe eltolásával vagy elforgatásával származtathatók. Ez a görbe tekinthető a primitív egyik paraméterének. A 2D ablakban azonosítással kiválasztható. További paraméterek: az eltolás ill. elforgatás hossza ill. szöge és irányítása. Ezzel a primitív a saját koordinátarendszerében definiálva van, de még el kell helyezni a világkoordinátarendszerben, vagyis pozicionálni kell. Ez történhet abszolút módon, koordináták (origó és irányvektor) megadásával, vagy relatív módon, egy már létező primitívhez való illesztéssel. Az utóbbit a 3D ablakban lehet azonosítani. Ahogy egymás után definiáljuk és pozicionáljuk a primitíveket, azok drótvázképe sorra megjelenik a 3D ablakban a megadott nézetnek megfelelően. Miután definiáltuk az összes primitívet, különböző 3D transzformációkat alkalmazhatunk rájuk: eltolás, forgatás, tükrözés, nagyítás. Mindez történhet mozgatással vagy másolással. Ezután következik a primitívek összerendelése: Itt adjuk meg, hogy a primitívek milyen sorrendben és milyen halmazműveletek által építik fel a tárgyat. Ezzel a tervezési folyamat végetért. A tárgy modelljét (CSG) a rendszer a 3D adatbázisban tárolja, így az később előhívható. A kész tárgyról a felhasználó különféle rajzokat kérhet (takartvonalas, takartfelületes, műszaki). Ezek tárolhatók a 2D adatbázisban és 2D-ben további rajzi elemekkel (méretvonalak, szöveg, stb.) kiegészíthetők, majd plotterrel kirajzolhatók. Lekérdezhetők a tárgy volumetrikus jellemzői is (térfogat, felszín, stb.). A 3D alrendszer főmenüje a következő:

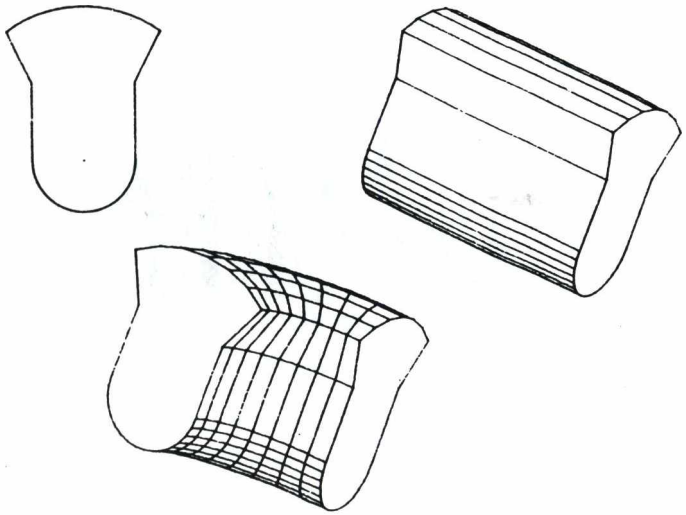
- **File:** 2D adatbázis kiválasztása, kilépés
- **Create:** primitívek definiálása és pozicionálása
- **Delete:** primitívek törlése
- **Transform:** primitívek 3D transzformációja (eltolás, forgatás, tükrözés, nagyítás, duplikálás)
- **Query:** a primitívek vagy a tárgy lekérdezése
- **View:** nézet kijelölés
- **Zoom:** zoomolás
- **Draw:** rajzgenerálás

Irodalomjegyzék

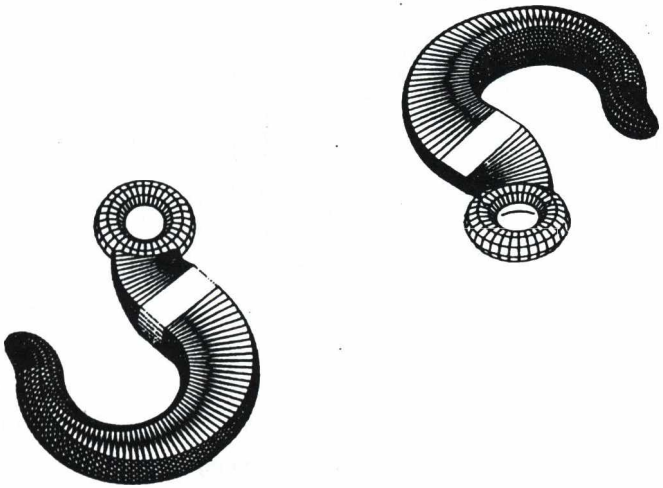
[1] Martti Mäntylä : An Introduction to Solid Modelling (Helsinki University of Technology, 1984)

[2] Szél János: XDMS - CAD adatkezelő rendszer (Mérés és Automatika 34/4 1986)

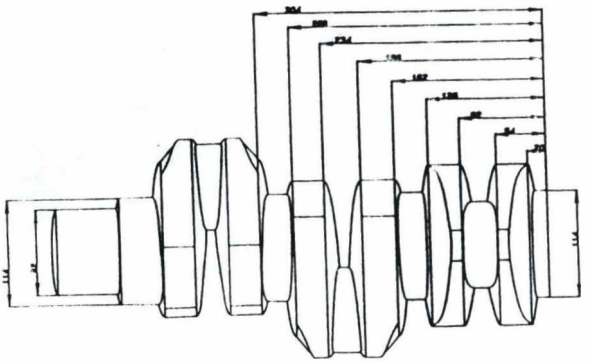
[3] Pikler Gy.: Dialógussal vezérelt interaktiv gépészeti CAD rendszerek elméleti és gyakorlati megfogalmazása. (MTA SZTAKI tanulmány 142/1983)



1. ábra: sweeping primitívek



2. ábra: horogszerü alkatrész



3. ábra: forgattyús tengely

UNIX-szal az oktatásban a BME Villamosmérnöki Karán

Máray Tamás, Szeberényi Imre
<maray@fsz.bme.hu>, <szebi@fsz.bme.hu>
BME Folyamatszabályozási Tanszék

A Budapesti Műszaki Egyetem Villamosmérnöki Karán hosszabb ideje használnak UNIX rendszereket az oktatásban illetve a kutatásban. A nyolcvanas évek elejére hazánkba is eljutottak különböző utakon UNIX változatok. Ezekkel kutatásokat folytatott több intézmény és vállalat is.

1982-ben a Budapesti Műszaki Egyetem Folyamatszabályozási Tanszéke is szert tett egy igen korai UNIX verzióra (V6, ez az első publikus UNIX változat), amit megpróbáltak az oktatásban alkalmazni. 1983-ra sikerült helyi fejlesztésekkel, módosításokkal kialakítani egy olyan rendszert, amit már viszonylag sok hallgató használhatott. A gépi hátteret ekkor a Villamosmérnöki Kar Számítástechnikai Csoportja által üzemeltetett SZM-4 számítógép jelentette, aminek az üzemeltése komoly feladatot jelentett, de a nehézségek ellenére sikerült 16 terminállal kb. 150 hallgatót kiszolgálni. (A gép 256 KByte operatív tárral és kb. 40 MByte cserélhető diszk-kapacitással rendelkezett.)

A kezdeti időkben a UNIX nem mint operációs rendszer került be az oktatásba, hanem mint eszköz. A villamosmérnök-képzés alapozó tárgyai között volt egy korábban gépi számítástechnikának, majd később számítógépek programozásának nevezett tárgy, melynek keretében programozási nyelveket tanultak a hallgatók. Alaptárgy lévén nagy létszámot kellett egyszerre oktatni, és ehhez gépi háttérként egy SZM-4 típusú gép állt rendelkezésre. Sajnos tapasztalataink azt bizonyították, hogy ez a gép az akkor elterjedt RSX-11 operációs rendszerrel nem igen tudott 4-5 interaktív terminálnál többet kiszolgálni, még akkor sem, ha a hallgatók csupán primitív, nem számításigényes házi feladat programokat fejlesztettek ill. futtattak. Ezért másik operációs rendszert kerestek az akkori számítóközpont szakemberei. Így jutott el egy kis csoport Kőrösi István vezetésével a UNIX-hoz, amellyel igen kedvező tapasztalataink voltak. Meglepő módon a rendszer

símán elvitt 14-16 terminált, amelyen hallgatók dolgoztak. Miért volt képes erre a UNIX ? Véleményünk szerint több tényező együttes hatásából adódott ez a különbség:

1. Maga a UNIX kernel jóval kisebb, mint az RSX kernele. A V6-os UNIX kb 30- 40 KByte méretű csupán. Ez annak köszönhető, hogy igen egyszerűen épül fel, és csak a lényeges rendszerfeladatokat (file-kezelés, processz-kezelés, processzek közötti kommunikáció) oldották meg benne. Ehhez a mérethez viszonyítva óriásnak tűnik egy mai kernel mérete (300-400KByte), de még ma is elmondható, hogy a kernel csak a lényegi dolgokat tartalmazza, és igen sok funkció - ami más rendszerekben a kernel része - USER szinten futó privilégizált processz segítségével valósítanak meg. (Ilyen pl. a "ps" program, ami a /dev/kmem-en keresztül "túrkál" a kernelben.) Természetesen a rendszer kis méretét számos más ügyes fogás is biztosítja, amelyeket ma már szerte a világon különböző rendszerprogramozói kurzusokon oktatnak is. (Pl. a rendszer belső szinkronizációs problémáinak megoldási módszere.)

2. UNIX-ban lehetőség van ún. osztott programszegmensű programok futtatására, ez azt jelenti, hogy az ilyen programok programkódja csak egy példanyban van a tárban. Így ha pl. egy editor programot egyszerre több felhasználó is használ egyidőben, akkor az editor program kódja csupán egy példányban van jelen a memóriában. Oktatási környezetben igen gyakori, hogy egy programot egyszerre többen használnak. Tekintve az adott gép igen kicsi memóriáját, az osztott programszegmens használata jelentős erőforrás megtakarítást jelent.

3. A rendszerbe beépített cache buffer lényegesen csökkenti a diszk műveletek számát.

Az akkori, viszonylag korai UNIX verzió természetesen némi bővítésre szorult, hiszen pl. csak 256 felhasználót tudott nyilvántartani, és az igények ezt kicsivel meghaladták. Ugyanis az évente jelentkező kb. 150 új hallgató mellett tudományos munkát végző hallgatók és oktatók is használták a gépet, így kb. 250-300 felhasználó volt regisztrálva szinte folyamatosan. Mivel a file rendszeren nem akartunk változtatni (habár ekkor már ismert volt számunkra is a 7-es verzió módosított diszk struktúrája, ami ezt a kérdést is érintette),

ezért kénytelenek voltunk a 6-os verziónál még csak byte-ban tárolt UID és GID használatát ill. értelmezését módosítani. Ennek lényege az volt, hogy egy felhasználót nem csupán az UID-je azonosított, hanem az UID és a GID együtt. Ugyanakkor a GID eredeti szerepét is némileg megtartottuk azzal, hogy továbbra is létezett csoport, amit a GID határozott meg. Ezen felül minden csoportban volt egy kitüntetett UID (a nullás), aki a csoporton belül korlátlan jogokkal rendelkezett, tehát létrejött a group super user. Ez kitűnően illeszkedett az oktatáshoz, hiszen a csoportokat a tanulókörök alkották, és a tanulókör oktatója volt a kitüntetett user a csoporton belül.

Más módosítások is szükségessé váltak ahhoz, hogy az akkor rendelkezésünkre álló UNIX az oktatásban jól hasznosítható legyen. Ilyen módosítás volt egy diszk-kvóta rendszer kialakítása, hiszen a felhasználók nagy száma és a gép kis diszk kapacitása ezt feltétlenül indokolta. Mi a login-logout procedúrára építettünk, illetve az ezekhez tartozó programokat cseréltük le ill. módosítottuk (init, getty). Minden felhasználóhoz a password file-ban lehetett diszk-kvótát rendelni, amit a logout pillanatában ellenőrzött a rendszer. Kvótatúllépés esetén nem lehetett elhagyni a rendszert. Ehelyett egy menü jelent meg, ami segítette a felhasználót abban, hogy a felesleges file-okat letörölje. Természetesen ekkor akár haza is mehetett az illető felhasználó, de ilyen esetben azon a terminálon nem tudott a következő hallgató dolgozni, aki viszont élve a jogával bármit törölhetett annak érdekében, hogy az előző felhasználót kiléptesse, hiszen már az ő ideje ketyegett. Talán kicsit durva módszernek tűnik, de a több éves tapasztalat azt igazolta, hogy mindig rend volt ezen a téren, és valójában az a bizonyos törlés, amikor más törölte ki valakinek a file-jait, nem következett be.

Részben az oktatási környezet, részben a viszonylag nagy felhasználói létszám mellett a kevés munkahely szükségessé tette, hogy a terminálidőt gépi úton foglalhassák a hallgatók. Ezért egy terminálidő-foglaló programot is készítettünk, amivel sikerült az adminisztrációból adódó tévedéseket minimalizálni. Az időfoglaló program nyilvántartotta, a felhasználó idő kvótáját, a már felhasznált időt és a gép üzemeltetési rendjét , és ennek megfelelően engedélyezte az időfoglalást.

További gondot jelentett az, hogy nem állt rendelkezésünkre egy egyszerű könnyen kezelhető, képernyő orientált szövegszerkesztő, amelynek a kezelését a hallgatók gyorsan elsajátíthatták volna. Ezért egy ilyen editor program fejlesztésére

is sor került. Ennek eredményeként sikerült kifejleszteni egy olyan editort, ami úgy gondoljuk nagy mértékben növelte a rendszer használhatóságát. Érdekes mellékhatást figyelhettünk meg, ugyanis az új editor használatával lényegesen csökkent a papírfelhasználásunk. Ennek oka az volt, hogy a kifejlesztett szövegszerkesztő segítségével egyidőben lehetett megjeleníteni a program forrását és a lista file-t is. Így nagyon sok esetben feleslegessé vált, hogy programfejlesztés közben a hibalistát kinyomtassa a programozó.

1983 és 89 között minden műszer szakos villamosmérnök hallgató (évi kb. 130-150 fő) legalább felhasználói szinten megismerkedett a UNIX-al. Persze minden évben voltak egy páran akik mélyebbre ástak az operációs rendszerben és így szép TDK munkák, diplomatervek születtek a témában. 1989-től a "számítógépek programozása" c. tárgy oktatását áthelyezték a kar PC laboratóriumába, megszüntetve ezzel az alapoktatást a UNIX operációs rendszeren. E döntésnek többek között az is oka volt hogy az akkor UNIX-ot futtató SZM-4 és TPA11/440 típusú gép teljesítménye már nem bizonyult elegendőnek, és nagyobb UNIX-os gép beszerzése márcsak az embargó miatt sem volt lehetséges. Másfelől viszont az is igaz, hogy a kar vezetése fontosabbnak ítélte az MS-DOS megismertetését a UNIX-nál.

1989-től tehát a mai napig, felsőbb évfolyamos hallgatók választható tantárgyak keretében, témalabor, önálló tanszéki gyakorlat útján vagy fakultatív módon ismerkedhetnek a UNIX-al. Körülbelül két éve vált általánossá a felismerés hogy a UNIX-ra egyre nagyobb szükség van, így a Villamoskar különböző tanszékein ma már egyre nagyobb számban (kb. 40) találhatók különböző UNIX installációk, kezdve az SCO PC-s rendszereitől a nagyteljesítményű munkaállomásokon át (SUN, IBM) a VAX-okon futó UNIX-okig.

A UNIX operációs rendszer kiválónak bizonyult oktathatóság szempontjából is. Rajta keresztül a hallgatók olyan általános fogalmakkal ismerkedhetnek meg mint pl. időosztásos rendszer, ütemezés, hierarchikus file-rendszer, hozzáférési jogok stb. Olyan nem kifejezetten a UNIX oktatását célul kitűző tárgyak mint pl. "rendszerprogramozás" v. "operációs rendszerek" is esettanulmányként a UNIX-ot mutatják be. Igen fontos, hogy a UNIX forrásban hozzáférhető, a kernel algoritmusok publikáltak, egyszerűek és jól megérthetők, és csaknem az egész rendszer magasszintű nyelven, tiszta, világos programozási stílusban íródott.

Az egyetem egyéb karain is kezd elterjedni a UNIX, ahol elsősorban mint

segédeszközt ismerik meg a hallgatók a különböző, UNIX-ra íródott alkalmazások (CAD stb.) futtatása során.

Jelenleg a BME számítógéphálózatában a "nameserver", "timeserver" és részben a "mail gateway" funkciókat is UNIX-os gépek látják el. Ezek konfigurálásában és üzemeltetésében villamosmérnök hallgatók is aktívan részt vesznek.

A UNIX grafikus felhasználói felülete az X-Window egyre nagyobb teret hódít, sok hallgató ismerkedik vele egyelőre fakultatív módon. Néhányan már diplomatervük témájává is választották. Sajnos ezen a területen az eszközhiány még korlátozó tényező. Ismerve az oktatási intézmények anyagi helyzetét, saját erőből gyors változás nem várható ezen a téren. Egyes profitorientált cégek azonban már felismerték azt hogy az egyetem támogatása hosszabb távon jól megtérül számukra. Így pl. a DEC support centert hozott létre, a SUN oktatóközpontot alakít ki az egyetemen, stb.

A Villamosmérnöki Karon jelenleg zajlik a tanterv reformja. Az új tantervben mind az informatika szakon, mind pedig a villamosmérnök-képzésben nagyobb szerepet kapott a UNIX. Már nem csupán fakultatív tárgyak keretében ismerkedhetnek a hallgatók a UNIX felépítésével, belső szerkezetével, hanem órarendi keretek között is. Alkalmazói szmbontból a különböző tervező és szakértői redszerek, valamint a korszerű adatbáziskezelés oktatását tűzte ki célul az új tanterv. Természetesen ennek egyre növekvő eszköz igénye van, igy közelebbi terveink is elsősorban erre irányulnak. Szeretnénk beszerezni új munkaállomásokat, X-terminálokat, nagyobb teljesítményű számítógépeket, és ezek üzemeltetéséhez szükséges szoftvereket.

1. *Kőrösi I., Szabó Z., Ketler I. – Bevezetés az SZM-4 számítógép UNIX operációs rendszerébe, BME VSZCS közlemény 1983 július*
2. *Szabó Z., Juhász F., Ketler I. – UNIX editor, BME VSZCS 1983 január*
3. *Kőrösi I., Ketler I. – UNIX Fortran az SZM-4 számítógépen, BME VSZCS 1982 augusztus*
4. *Kőrösi I., Ketler I. – MACRO-11 Assembler a UNIX operációs rendszerben, BME VSZCS 1983 január*
5. *Szeberényi I., Forgács P., Ketler I. – NBS PASCAL a UNIX operációs rendszerben, BME VSZCS 1984 február*
6. *Máray T., Szeberényi I. – A nyíltság nyitja, Heti Chip 1992 április*

III.szekció: **OSZTOTT és KONKURRENS ELÉRÉSI TECHNIKÁK**

Osztott funkciók implementálása

Vincellér Zoltán, Porkoláb Zoltán, Csizmazia Balázs
ELTE Általános Számítástudományi Tanszék

Ezen ismertető keretében szeretnénk bemutatni a UNIX rendszerrel kapcsolatos decentralizálási kísérletek eredményeit és eszközeit. Az alapokat képező hálózati transzport réteg elérése után szó lesz sok vonatkozásban a távoli eljáráshívásokról és az osztott környezetben (részben a távoli eljáráshívással kapcsolatban) felmerülő biztonsági kérdésekről.

A Berkeley Socket Interface

Az 1970-es évek végén a Berkeley Egyetemen fejlesztéseket folytattak a UNIX operációs rendszer olyan kibővítésével kapcsolatban, amely lehetővé tette az Internet hálózat elérését felhasználói programok részére. Az eredmény – amelyet ma Berkeley **socket interface** és IPC (Inter Process Communication) néven ismerünk, és jelen formájában a BSD (Berkeley Software Distribution) 4.1c verzióban jelent meg – sokkal többet valosított meg. A socket interface képes biztosítani portábilis hálózati alkalmazások megírását, függetlenül a kommunikáció alapját képező, alsóbbszintű rétegektől. Az IPC és a socket interface azóta kvázi-szabvánnyá vált, és amellett, hogy a UNIX hálózati alkalmazások nagyobb részét kitevő Internet csatlakozások alapja, éppugy használják DECnet, X.25 és egyéb protokollokhoz. (Magukat a socket-interface funkciókat kernelhivásokként implementálták, eltérően a felhasználói könyvtárként létrehozott XTL interface-től.)

A kommunikáció – ahogy azt multi-taszk környezetben elvárjuk – nem gépgép (illetve hálózati node-ok közötti), hanem processz - processz szintű. Mivel egy process egyidejűleg több, különböző hálózati kapcsolatot is fenntarthat, ezért szükség van egyedi hálózati végpontok definiálására. Ezek az absztrakciók - a socketok - a pontok, ahonnan üzeneteket küldhetünk ill. ahol fogadhatjuk őket. A socketokat – akárcsak a filerendszerbeli fileokat – dinamikusan kreálhatjuk. Egy descriptor tér vissza, amelyet a felhasználói program az új socket azonosítására használ. A BSD rendszere más tekintetben is szorosan követi a UNIX filerendszert: ahol lehetséges a socketok pontosan úgy viselkednek mint a fileok vagy deviceok, pl. alkalmazhatjuk rájuk a hagyományos read(2) és write(2) kernelhívásokat.

Természetesen a hálózati I/O jóval komplexebb, mint a file I/O. A hálózati kommunikációban részt vevő processzek legtöbbször nem azonos szerepűek: egy tipikus **client-server kapcsolat** nem szimmetrikus, külön "menetrendje" van a kapcsolat felépítésének a két esetben. Szervernek szokás azokat a processzeket nevezni, amelyek valamilyen szolgáltatást nyújtanak más pro-

cesszeknek a hálózaton keresztül. Egy processzt akkor neveznek kliensnek, ha valamilyen szerverhez fordul valamilyen ügyben, és vár a szerver válaszára. Természetesen a kliens-szerver modell mellett léteznek más típusú viszonyok is, például az **információ-terjesztés** (ld. Comer, 1987), de ezeket az algoritmusokat jelenleg kevesebb helyen alkalmazzák.

A kapcsolat maga is kétféle lehet: **connection-oriented** vagy **connectionless**. Az első esetben egy virtuális csatorna épül ki a kommunikáló végpontok között, adatáram-jellegű, viszonylag hosszú életű kapcsolatok számára. A második esetben ellenben önállóan is értelmezhető üzeneteket továbbítunk (datagrammák), leggyakrabban gyors reakció-idejű, rövid kommunikációban.

Mindkét kapcsolati mód a szerver és kliens oldalon is maguknak a socketoknak a létrehozásával kezdődik, a socket() hívással. A híváskor specifikáljuk a használt címzési típust, a kapcsolat módját, és esteleg a defaulttól eltérő protokollt is (ez lehet például TCP vagy UDP). A visszatérő érték a socket-leíró, hasonló a UNIX file descriptorhoz. A létrehozott socket most még csak egy kapu a helyi transzport réteg felé, hálózati jelentősége akkor keletkezik, amikor egy nevet (címet) rendelünk hozzá a bind() hívással. Elvileg a bind() opcionális, a rendszer automatikusan is megteszi, generált egyedi névvel. Ez a legtöbb esetben meg is felel a kliens-oldali processzeknek, mivel ez a név csak annyira fontos, hogy ide küldi válaszát a szerver. A szerver oldal ugyanakkor köteles gondoskodni arról, hogy a kapcsolatot kezdeményező kliensek megtalálják. Ennek érdekében saját, a kliensek számára is "jól ismert" nevét rendeli a sockethoz. Ez például egy Internet TELNET szerver esetében a gép internet címéből és a TELNET megállapodás szerinti "jól ismert" port-számából (23) áll össze.

Ezután a szerver és kliens feladatai szétválnak. A szerver a listen() hívással várakozik arra, hogy kapcsolatot keressenek vele, ha ez bekövetkezik, az accept() segítségével határozza meg az őt megszólító processz nevét (azt amit a kliens explicit vagy implicit bind-olt). A kliens ezt a hívást a connect() funkcióval kezdeményezi, ahol specifikálja a felhívandó szerver "jól ismert" nevét. Az accept() létrehoz egy új socket-ot, amely csak a most létrejött kapcsolatot szolgálja, (és egyben lehetővé teszi, hogy a szerver az eredeti socket-on további klienseket fogadjon). A szerver processz ilyenkor általaban fork()-ol, a szülő tovább várakozik, a gyerek pedig exec()-el az adott klienst kiszolgáló peer-processzre. Az ilyen típusú szervereket szokás párhuzamos szervereknek nevezni.

Az adatcsere a read(), write(), send(), recv(), sendmsg(), recvmsg() hívásokkal történik. Mindegyik első paramétere (a file-descriptor helyén) a socket-leíró. A négy utóbbi rutin annyival több a read() és write() hívásnál, hogy további flag-ekkel szabályozhatjuk az átvitelt. A tovább már nem használt socket-okat close() vagy shutdown() hívással szabadíthatjuk fel.

A connectionless adatatkapcsolat kissé eltér a fent leírttól. A listen()-connect()-accept() hívássorozat két processz között virtuális csatornát épített ki, ahol az adatcsere lezajlott. Ehelyett magukban az üzenetekben is specifikálhatjuk a címzettet. Erre a sendto() es recvfrom() hívásokat használhatjuk, más-más processznek címezve datagrammjainkat. Ha azonban

azonos címzettnek küldünk több üzenetet, itt is használhatjuk a connect()-et anélkül, hogy listen()-elő processz várná üzenetünket.

A socket-interfacenél kell megemlíteni a select() hívást, bár mint általános esemény-kezelő rutin nem-hálózati alkalmazásoknál is használatos. Ha egy processz egy socket-ot olvasni próbál, a read() rutin blokkolja az I/O befejeztéig. Így gondokat okoz több socket párhuzamos figyelése, illetve time-out programozása. A select() hívással specifikálhatjuk azokat az eseményeket (socket olvasásra/írásra kész, időtúllépés, stb ...), amelyek bekövetkeztéig blokkolni akarjuk processzünket.

Egy connection-oriented szerver processz vázlata:

```
#include  <sys/socket.h>

int  s,  ns;
int  queue_size;
struct  sockaddr  serv_addr,  client_addr;

if  ((s  =  socket(AF_INET,  SOCK_STREAM,  0))  <  0)
{
        /*  socket-hiba  */
}
else  if  (bind(s,  &serv_addr,  sizeof(serv_addr))  <  0)
{
        /*  bind-hiba       */
}
else  if  (listen(s,  queue_size)  <  0)
{
        /*  listen-hiba  */
}

for  (  ;  ;  )
{
        if  ((ns=accept(s,  &client_addr,sizeof(client_addr)))  <  0)
        {
                /*     accept-hiba     */
        }
        else
        {
                if  (fork()  ==  0)       /*     gyerek     */
                {
                        close(s);
                        solve_peer_client_problem(ns);
                        exit(0);
                }
                else                                  /*     szulo     */
                {
```

```
                close(ns);
            }
        }
}
```

és a fenti szerver kliens párja:

```
#include <sys/socket.h>

int s;
struct sockaddr serv_addr, client_addr;

if ((s = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
        /* socket-hiba */
}
else if (bind(s, &client_addr, sizeof(client_addr)) < 0)
{
        /* bind-hiba      */
}
else if (connect(s, &serv_addr, sizeof(serv_addr)) < 0)
{
        /* connect-hiba */
}
else
{
        req_to_server(s);
        close(s);
}
```

Az X/Open Transport Interface

Az X/Open Group 1984 óta jelenteti meg X/Open Portability Guide néve kiadványait. Maga az X/Open csoport nem szabványosító szervezet. Az üzle világ nagyjainak közös vállalkozásaként már létező szabványokat emelnek ki, é alkotnak konzisztens környezetet, operációs rendszer funkcióktól egészen prog ramozási nyelvekig. Ahol még nem letezik hivatalos szabvány ott a de fact standardot veszik figyelembe. Az X/Open csoport tagjai garanciát vállalnak kiválasztott szabványok támogatására.

Az X/Open Portability Guide első két kiadása öt fejezetből állt. Ezek so rendben: Operációs rendszer parancsok és utilityk (System Commands). Prog ramozói interface (Kernelhívások). Supplementary Definitions. Programnyel vek (C és COBOL). Adatkezelés (C-ISAM és SQL). Az 1988-as harmadi

kiadás kiegészült két fontos témakörrel: Grafikus környezet (X-Windows) és Hálózati szolgáltatások (XTI).

Az X/Open Transport Interface (XTI) az AT&T UNIX Transport Layer Interface (TLI) definícióján alapul. Az ISO OSI 7 rétegű hálózati referencia modelljében a (4.) transzport réteg különösen fontos szerepet tölt be. Ez a legalsóbb réteg amely végpontok közötti megbízható adatcserét biztosít. Másrészt ez a réteg definiál számos olyan szolgáltatást, amelyek közösek a különböző hálózati protokollokban, beleértve az ISO protokollokat, a TCP/IP-t, az SNA-t, stb. A fentiek miatt a transport layer interface képes arra, hogy a felhasználói programok ill. felsőbb szintű protokollok elől "eltakarja" az alsobb szintek specialitásait. Az XTL így lehetővé teszi mind protokoll, mind médium-független hálózati elérést programok illetve a felsőbb protokollok számára. A TLI elsődlegesen az OSI transzport szerviz definíciói (az OSI Connection-oriented és Connectionless szervízek) alapján keszült, de a gyakorlatban kibővítették más protokollok támogatására is, mint a TCP, UDP, DECnet, SNA, Xerox Network. Ugyanakkor az XTL szabvány definíciója csak az ISO, TCP és UDP protokollokat említi.

Az AT&T TLI használói tapasztalhatnak eltéréseket az XTL alkalmazása során. Ilyen bizonyos függvények kibővített hibakódja, újabb event-ek bevezetése, melyek az aszinkron végrehajtást támogatják, néhány a TCP-t támogató eszköz a t_send(), t_sndrel() és t_rcvrel() függvényekben, stb. A változtatások a TCP nagyobb támogatását és a hordozhatóságot szolgálják. Habár az XTL, mint absztrakt szabvány, nem rendelkezik az implementációról, azt rendszerint az AT&T TLI-hez hasonlóan a STREAMS-re épülő felhasználói könyvtárként valósítják meg.

Az XTL természetesen sokban hasonlít a Berkeley Software Distributions socket-alapú IPC-jére. Mindkettő az alsóbb transzport szolgáltatások elérését biztosító programozói interface. Mindkettő file descriptort használ a hálózati végpontok azonosítására. Vannak azonban jelentős eltérések is:

- Az XTL biztosít olyan függvényeket, amelyek segítségével kapcsolat kiépítésekor meghatározhatjuk az aktuális transzport réteg típusát, rákérdezhetünk illetve beállíthatunk protokol opciókat. Szelektíven választhatunk kapcsolatkiépítést több beérkező igény között.

- Az XTL lehetővé teszi, hogy ugyanazt a végpontot több processz között osszuk meg, például fork() és exec() rendszerhívások után.

- A kapcsolat elengedésekor az XTL program egy felhasználó által definiált disconnect üzenetet küldhet, amiben közölheti például a kapcsolatfelbontása okát, nyugtát küldhet, stb.

Távoli eljárás-hívás (RPC)

Az RPC (Remote Procedure Call) a hálózati osztott rendszerekben klasszikusnak számító modell, amely ugyan nem mindenben felel meg az OSI hivatkozási modell előírásainak, mégis a legtöbb UNIX rendszerben implementálva van. Mivel a gyorsaság erősebb szempont volt a létrehozásakor, mint a szabványokhoz való igazodás, ezért nem is réteges szerkezetű, és a párbeszéd menedzselésére valamint a hibakezelésre is egyedi, összeköttetésmentes protokollt használ.

A UNIX rendszerekben használt távoli eljárás-hívás a következő lépésekben valósul meg: az ügyfél program (kliens) meghívja a hozzászerkesztett ún. **csonk-eljárást** (1. lépés). A hívó eljárás számára a hívás ugyanolyan, mintha egy helyi hívás volna. Az ügyfél csonk átveszi a paramétereket, és bepakolja egy üzenetcsomagba (**paraméter-rendezés**), és átadja a kernel segítségével a transzport interfésznek (2. lépés). Innen az adatok az adott hálózatra jellemző módon továbbítódnak a szolgáltató gép (szerver) transzport entitásához (3. lépés), amely azokat átadja a szolgáltató-programhoz szerkesztett **szolgáltató csonknak** (4. lépés).

A szolgáltató csonk kiválogatja a hívási paramétereket, és ezekkel meghívja a szolgáltató eljárást (5. lépés). (A szolgáltató eljárás sem érzékeli hogy távolról hívták, hiszen közvetlen hívója egy helyi eljárás.) A szolgáltató eljárás elvégezve munkáját visszatér az őt hívó szolgáltató csonkba (6. lépés), amely az eredményt egy üzenetcsomagba rendezve egy rendszerhívással átadja a szolgáltató gép transzport interfész programjának (7. lépés). Miután a válasz visszaérkezik az ügyfélgép transzport szintjéhez (8. lépés), közvetlenül az ügyfél csonkhoz kerül (9. lépés). Végül az ügyfél csonkból visszatér az ügyfél eljáráshoz (10. lépés).

A hívó program számára a **transzparenciát**, vagyis azt hogy a helyi és a távoli hívások között ne kelljen különbséget tennie, meglehetősen nehéz biztosítani. A legnagyobb problémát a paraméterek átadása jelenti.

Elemi típusok és strukturák érték szerinti átadásánál csak az adott tipus hálózati ábrázolásra történő konvertálása okoz némi teendőt. A probléma ott kezdődik, amikor hivatkozás szerinti (azaz cím szerinti) paraméterátadást kell megoldani. A hívott eljárás tudja, ha hivatkozási paramétereket kezel, ezért mutatóműveleteket végez velük. Távoli hívásra ez nem működik. Amikor a szerkesztőprogram generálja a szolgáltató programot nem tudhat az RPC-ről, és a szokásos utasításokat állítja elő a mutatók kezelésére. A kapott paraméterben kijelölt objektum még csak nem is a szolgáltatógépen van általában, de ha ott lenne sem lehetne ugyanaz a címe, mint ami az ügyfél programban volt.

A megoldás egy lehetséges alternatívája a másolás/visszaállítás szerinti hívás. Ekkor az ügyfél csonk lokalizálja a mutató által kijelölt objektumot, és átadja a szolgáltató csonknak. A szolgáltató csonk elhelyezi azt valahová a memóriájában, és a szolgáltató eljárásnak egy erre mutató pointert ad át, így a szolgáltató a szokásos módon éri el az eredeti adat másolatát. Amikor a szolgáltató eljárás visszatér a csonkba, az a valószínüleg módosított adattételt

is visszaküldi az ügyfél csonknak, amely azt az eredeti hivatkozási paraméter felülírására használja. Bár a **másolás/visszaállítás** mechanizmus általában jól használható, bizonyos esetekben hibákat okoz. Nézzük a következő program részletet:

```
main()
{
    int  a=0;

    duplainkr(&a,&a);
    printf("%d",a);
}

duplainkr(x,y)
    int  *x;
    int  *y;
{
    (*x)++;
    (*y)++;
}
```

Helyi futás eredményeként a main függvénybeli "a" változó értéke 2-vel növekszik, a kinyomtatott szám a 2-es. Ha a duplainkr eljárást **másolás/visszaállítás** szerint RPC-vel hívjuk meg, az ügyfél csonk az "a" változó két másolatát küldi át a szolgáltató csonknak, ahol mindkét másolat inkrementálódik, majd visszakerül az ügyfél csonkhoz, és az sorban visszaállítja azokat. Először, majd másodszor is 1-nek írja felül az "a" eredeti 0 értékét, a kinyomtatott szám tehát 1 lesz, a helyes 2 helyett.

A mutatók különösen problémásak, ha egy összetett lista, gráf, vagy egy bonyolult adatstruktúra közepébe mutatnak. Az eljárás-, és a függvényparamétereket ugyancsak nehéz kezelni, bár a szolgáltató csonk olyan - a szolgáltató gépre nézve lokális - eljárásokkal helyettesítheti azokat, amelyek fordított irányú RPC-t használva visszahívhatják a hívott eljárásokat az ügyfél gépen. Sok RPC-rendszer úgy oldja meg ezeket a problémákat, hogy távoli hívások esetén megtiltja a hivatkozási paraméterek, mutatók, az eljárás- ill. függvényparaméterek használatát. Ez nagyban megkönnyíti az implementálást, de egyben lemondást jelent a transzparenciáról, mivel a helyi és a távoli hívások formálisan is különbözni fognak.

Egy másik lényeges kérdés az, hogy honnan tudja az ügyfél csonk, hogy mit kell hívnia? Erre egy egyszerű, de nagyon dinamikus mechanizmus használható, amelyet Birrel és Nelson (1984) javasolt. Ennek lényege az, hogy amikor egy szolgáltató program elindul egy szabvány üzenet elküldésével bejegyezteti magát egy adatbázisba. Az üzenet tartalmazza a szolgáltató nevét (ASCII string), a hálózati címét, és egy egyedi azonosítót (pl. egy 32-bites véletlen számot). Amikor egy ügyfél program végrehajtja az első hívását, az ügyfél

csonk a szolgáltató nevével az adatbázishoz fordul és visszakapja annak hálózati címét valamint egyedi azonosítóját. Ez a folyamat a **kötés** (binding). Az egyedi azonosítót minden egyes RPC hívásnak tartalmaznia kell, a szolgáltató gép transzport interfésze ez alapján dönti el, hogy a kapott üzenetet melyik szolgáltató csonkhoz kell továbbítania.
Van egy másik fontos szerepe az egyedi azonosítónak. Ha a szolgáltató összeomlik, majd újraindul, akkor egy új egyedi azonosítóval jegyezteti be magát az adatbázisba. Az ügyfelek régi azonosítóval végrehajtott hívásai sikertelenek lesznek, így értesülnek arról, hogy összeomlás történt, és újrakötést kell kezdeményezniük.

A Kerberos rendszer

A Kerberos rendszert az Massachusetts Institute of Technologyn fejlesztették ki az Athena projekt keretében. A projekt célja egy nagyon sok számítógépből álló (oktatási célú) hálózat kialakítása és a működéshez szükséges software elkészítése volt. A projektben a 4.3BSD UNIX rendszert használták és bővítették. A Kerberos célja az volt, hogy a hálózatba rakott "nem megbízható" számítógépekkel se lehessen a rendszerben illetéktelenül dolgozni, a szerver és a kliens közt áramló adatokat titkosítani lehessen, és **a kliensek ne tagadhassák le kilétüket a szerverek előtt.** A Kerberos rendszer a következőképpen működik: van egy belső adatbázis, amely (**azonosító, kulcs**) párokat tárol, ahol az azonosítók között előfordul az egész rendszerhez hozzáférő összes felhasználó azonosítója, és minden egyes szerverhez is tartozik egy-egy azonosító. Az azonosítóhoz tartozó kulcs az adott program saját kulcsa, amivel az üzeneteit titkosítja, és csak a Kerberos és maga a program ismeri azt.

- Amikor a felhasználó bejelentkezik valamelyik számítógépbe, és már beírta a nevét a `login:` prompt után, a login program egy üzenetet küld a Kerberos illetékesség-vizsgáló szervernek. Az üzenet két szöveget tartalmaz: az egyik szöveg a felhasználó azonosítója, a másik szöveg pedig az ún. **jegykiadó szerver** (ticket-granting szerver, TGS) nevét tartalmazza (e szerver szerepéről később lesz szó).

- Az illetékesség-vizsgáló szerver a fent említett adatbázisból kikeresi a kapott üzenetben lévő két azonosítóhoz tartozó kulcsokat (a felhasználói azonosítóhoz tárolt kulcs megegyezik a felhasználó titkosított jelszavával). Ezután a szerver egy válaszüzenetet küld vissza, ami tartalmaz egy titkos ún. TGS-kulcsot, és egy ún. **TGS-jegyet.** Ez a TGS-jegy tartalmazza a felhasználói azonosítót, a jegykiadó szerver (TGS) nevét, annak a számítógépnek az Internet címét, amelyen a felhasználó dolgozik, és a fent említett TGS-kulcsot. A TGS-jegy az üzenetben titkosítva van a felhasználó által nem ismert kulccsal (a jegykiadó szerver azonosítójához

tartozó kulccsal), és a teljes üzenet titkosítva van a felhasználó titkosított jelszavával, hogy más ne férjen hozzá a tartalmához.

- A felhasználó jelszavának beadása után az illetékesség-vizsgáló szervertől visszakapott üzenetet a login program a felhasználó titkosított jelszavával dekódolja, és a TGS-kulcsot valamint a (titkosított) TGS-jegyet eltárolja.

- Ezután ha a felhasználónak valamilyen hálózati szolgáltatásra van szüksége, akkor a jegykiadó szervertől (TGS-től) kérnie kell egy "jegyet" a kívánt szerverhez, és a szervernél ezzel a jeggyel azonosíthatja magát. Ennek menete a következő pontok témája.

- Ha például a nyomtató szervert akarjuk használni, akkor ahhoz egy jegyet kell kérni a jegykiadó szervertől. Ez úgy történik, hogy a jegykiadó szerverhez egy olyan üzenetet kell küldeni, amely tartalmazza a korábban eltárolt TGS-jegyet, a nyomtató szerver nevét és egy, a korábban megkapott TGS-kulccsal titkosított adatmezőt (ellenőrző mezőt). Ez az ellenőrző mező tartalmazza a felhasználó bejelentkezési nevét, a gépjének az Internet címét, és a pillanatnyi időt.

- A jegykiadó szerver a kapott üzenet ellenőrzése után válaszüzenetet küld, amiben van egy új kulcs, és egy új (titkosított) jegy. Ez a titkosított jegy a szolgáltatást kérő kilétét igazolja, és a nyomtató szerver kulcsával van titkosítva (ami a Kerberos adatbázisában van, és **csak a nyomtató szerver és a Kerberos ismeri**). A válaszüzenet a korábbi TGS-kulccsal van titkosítva.

- A felhasználó programja az üzenetet dekódolja, és a nyomtató szervernek a fenti "új jegyet" kell elküldenie (másik két adatmező mellett), ezzel igazolhatja kilétét. A másik két adatmezők közül az egyik a nyomtató szerver nevét, a másik pedig (a korábban említett ellenőrző mezőhöz hasonlóan) a felhasználó bejelentkezési nevét, a gépjének az Internet címét, és a pillanatnyi időt tartalmazza (ez a mező az előző pontban említett új kulccsal van titkosítva, ezért ezt a nyomtató szerver dekódolni tudja).

A két helyen is említett ellenőrző mező tartalmazza a pillanatnyi időt is. Ez azért lényeges, mert ezek a mezők viszonylag rövid élettartalmúak, nehogy valami illetéktelen "hálózatlehallgató" újra fel tudja használni. Viszont ekkor meg kell oldani azt, hogy a hálózatban levő gépek órái szinkronizálva legyenek. Ez a 4.3BSD UNIX rendszerben úgy van megoldva, hogy van egy idő-szerver gép, amely összegyűjti a hálózatba kapcsolt gépek órájának állását, majd az egyes gépeket utasítja arra, hogy az **órájuk sebességét** növeljék - csökkentsék ahhoz, hogy az "átlag-időt" elérjék.

Ajánlott irodalom

- Douglas E. Comer: Operating System Design
  Volume II - Internetworking with XINU
  Prentice-Hall Inc., 1987
- Andrew S. Tanenbaum: Modern Operating Systems
  Prentice-Hall Inc., 1992
- Andrew S. Tanenbaum: Computer Networks , 3rd Ed.
  Prentice-Hall Inc., 1988
- Birrel, A.D. és NELSON, B.J.: Implementing Remote Procedure Calls
  ACM Trans. on Computer Systems, vol. 2 (1984)
- Tay, B.H. és Ananda, A.L.: A Survey of Remote procedure Calls
  Operating Systems Review, vol. 24 (July/1990)

Elosztott rendszerek, elosztott file-rendszerek UNIX környezetben

Biczók László, KFKI Rt. XEUS Rendszerépítő Iroda

Túlzás nélkül állíthatjuk, hogy az utóbbi években a világon a mikroszámítógépes és "workstation" kategóriájú rendszerek körében igen népszerűvé váltak a különféle lokális hálózati architektúrák. A kliens-szerver architektúrájú helyi hálózatok és az egyre jobban terjedő intelligens terminál rendszerek megfelelő szervezés esetén igen nagy mértékben segíthetik a PC munkahelyek előtt helyet foglaló felhasználók munkáját, növelhetik annak hatékonyságát, csökkentve egyúttal az egy munkahelyre eső eszközigényt, ill. ezzel összefüggésben a költséget is.

Helyi hálózatok

Az ily módon felépített hálózati szoftverek lehetővé teszik, hogy a felhasználó hozzáférhessen a távoli szerver gép(ek) szolgáltatásaihoz, esetenként akár oly módon is, hogy a távoli erőforrások nem különböztethetők meg a helyiektől. A felhasználó ilyen esetekben ugyanolyan módon férhet hozzá a távoli erőforrásokhoz, mint a helyiekhez, anélkül, hogy új parancsokat, szintaktikát, ill. környezetet kellene megismernie. Hasonló szolgáltatást biztosít például a NOVELL, vagy a XEUS Intelligens Terminál Rendszer, a rendszer DOS módjában. Ezen szoftverek segítségével a felhasználó egy logikai drive-ként látja a szerver számítógép háttértárát hasonlóan ahhoz, ahogy a helyieket. Ez a megoldás igen hasznos, hiszen "kitárul a világ" a felhasználó előtt; a rendelkezésére álló erőforrások igen nagy mértékben megnövekednek.

Növelve azonban a rendszerek méretét, problémák jelentkeznek. A felhasználónak "fejben kell tartania" az egyes logikai drive-okon elhelyezkedő rendszerek tulajdonságait (kapacitás, adatfile-ok, file-ok elhelyezkedése, stb.), s ezáltal, ha közvetett módon is, tudomása van a

rendszer felépítéséről, arról, hogy melyik logikai drive melyik szerverhez tartozik, stb. Továbbá, amennyiben az egyik szerveren rendelkezésre álló diszk terület erősen lecsökken, egy másikra kell váltani (hasonlóan a 4.x előtti DOS verziók problémájához). A probléma megoldható új erőforrások hozzáadásával a szerveren, azonban ez csak rövid távú megoldás. Lehetséges ugyanis, hogy a hálózatban levő szerverek közül csak egy szenved a kevés erőforrás "betegségében", a többi bőven rendelkezik még velük. Ebben az esetben a "gyengélkedő" szerver bővítése pocsékolás, hiszen a hálózat összes erőforrása bőségesen elegendő lenne a felhasználók kiszolgálására. Természetesen a felmerülő kézenfekvő megoldás a rendszer megfelelő konfigurálása. Nagyobb rendszereknél azonban gyakran nem látható előre a pontos terheléseloszlás, az utólagos átkonfigurálás pedig esetenként igen nagy erőfeszítéseket követel, adott esetben jelentős többletkiadással együtt (újrakonfigurálási problémák, időkiesések, stb.). Elképzelhető még az, hogy a rendszerekbe utólagosan valamilyen automatikus terhelés elosztó funkció beépíthető, annak hatásfoka azonban, lévén hogy a rendszerek tervezésekor nem számoltak ilyen funkciókkal, megkérdőjelezhető, a vele járó teljesítmény csökkenés nehezen elviselhető.

Újabb problémát jelent a rendszer megbízhatósága. A szerver gép meghibásodása esetén a rajta lévő információ (adat, programok) időlegesen hozzáférhetetlenné válhatnak, komolyabb problémák (diszk sérülések, stb.) esetén véglegesen elveszhetnek a felhasználók számára. Az ezt kiküszöbölő megoldások (minden lemezművelet elvégzése egy másik lemezen is, stb.) komoly beruházásokkal járnak.

Intelligens Terminál Rendszerek

Intelligens terminál rendszerek alkalmazása esetén a fentebb említett problémák szintén jelentkeznek a rendszerek DOS módjában. Amennyiben a rendszerek UNIX terminál módját használják a felhasználók, akkor lényegében a szerver (gyakrabban használt terminológia szerint: host) gépen dolgoznak. Itt a korábbi problémák nem jelentkeznek, hiszen a felhasználó egy egységes file-rendszerben dolgozik, amelyben nem látja azt, hogy a file-

rendszer egyes részei hogyan helyezkednek el a host gép erőforrásain. A további bővíthetőség sem okoz komoly problémát, az egyes host gépek ugyanis összeköthetők, és megfelelő szoftver használata esetén (NFS, RFS, stb.) más host-ok file-rendszerei (ill. annak részei) beleépíthetők (mount) az éppen használt host file-rendszerébe. Ezzel a megoldással egy kényelmesen bővíthető rendszert lát a felhasználó maga előtt.

A megoldás azonban most sem problémamentes. A felhasználó ugyanis azzal, hogy PC-jét terminálként használja, saját erőforrásairól (vagy legalábbis azok egy részéről) lemond. Ezen a problémán sokat segíthet az, ha a PC-t intelligens terminálként használva a rendszer bizonyos feladatokat rábíz a terminálra. Ezzel a megoldással egyrészt tehermentesíti a host gép(ek)et, másrészt lényegesen jobban kihasználhatja a terminál kínálta lehetőségeket.

Ahhoz azonban, hogy a felhasználó teljes egészében élvezhesse mindazokat az előnyöket amiket egy ilyen rendszer nyújthat, a PC terminált annak DOS módjában is használnia kell. Ekkor azonban a felhasználó kénytelen két különböző operációs rendszer használatát megtanulni, a vonatkozó parancsokkal, szintaktikákkal együtt. Mivel a felhasználók jó része nem képes (vagy nem hajlandó) erre, a munka hatékonysága csökkenhet, munkaköri problémák léphetnek fel.

Másik probléma, hogy ez a rendszer sem küszöböli ki teljesen a bővítés problémáit. Az egyes könyvtárak, a benne létrehozott file-ok mérete nem növelhető tetszés szerint. Ezen könyvtárak ugyanis egy adott szerver háttértárán helyezkednek el, bővítésüknek az adott drive kapacitása szab határt. Így előállhat az a helyzet, hogy egy adott könyvtárat a felhasználó nem bővíthet tovább, míg mások gond nélkül használhatók továbbra is.

A megbízhatóság ugyan növekedhet intelligens terminálok használata esetén, teljes megoldást azonban nem nyújt ez a rendszer sem. A növekedés annak köszönhető, hogy a host-okat összekötő hálózati szoftverek egy része (pl. NFS) ún. "stateless" protokollt használ az egyes szolgáltatások eléréséhez. Ez esetben a szolgáltatást kiszolgáló host nem tart nyilván információt a szolgáltatást kérő gépről, miután befejezte a kérés kiszolgálását. Ha két szolgáltatás kiszolgálása között a kapcsolat valamilyen

okból megszakadna a két gép között, a host állapota teljesen konzisztens marad, nem áll fönn információvesztés esélye. Ha a kapcsolat újra helyreáll, az új kérések kiszolgálása probléma nélkül folytatódhat.

Diszk sérülések ellen azonban ebben az esetben is csak költséges megoldásokkal védekezhetnek a felhasználók.

Újabb problémát jelent az ún. "cascading mount", ami lényegében azt jelenti, hogy egy "mount"-olt távoli file-rendszer valamely alkönyvtárához egy harmadik host alkönyvtárrendszerét "mount"-olva az abban elhelyezkedő file-ok elérhetőségéhez szükséges a közbenső host elérésének lehetősége is. Ez azt jelenti, hogy a közbenső host "kiesése" esetén a harmadik gépen levő adatok is elérhetetlenné válnak, jóllehet maga a host képes lenne a hozzáférések kiszolgálására.

További problémát jelent a felhasználók mobilitása. Általános tapasztalat, hogy a felhasználók nem kötődnek kizárólag egy géphez, gyakran váltogathatják a helyet, ahonnan bejelentkeznek a rendszerbe. Természetes igény, hogy bármely terminálról bejelentkezve a felhasználó ugyanazt a környezetet lássa (home directory, hozzáférési jogok, környezeti változók, stb.). Ez esetenként csak bonyolult konfigurációk segítségével érhető el.

A különböző PC terminálok előtt ülő felhasználóknak időnként szüksége lehet arra, hogy egy másik terminál adataival dolgozzanak. A fentebb említett megoldások ezt a problémát vagy teljességében figyelmen kívül hagyják, vagy lehetővé teszik ugyan a másik terminál adataihoz való hozzáférést, ám új parancsok, konfigurációs paraméterek megtanulását várják el a felhasználóktól.

Elosztott rendszerek

A fentebb említett problémákra egy egységes, a felhasználó számára teljesen konzisztens megoldást nyújtanak az ún. elosztott rendszerek (Distributed Systems). Ezek a rendszerek egy egységes (és igen gyakran a rendszer által összeállított, virtuális) file-rendszert biztosítanak a felhasználók részére, amelyben azok szabadon mozoghatnak, az alkönyvtárakban

elhelyezkedő file-okhoz megfelelő (és egységes) védelmi rendszer mellett férhetnek hozzá anélkül, hogy a legcsekélyebb tudomásuk lenne a file-ok rendszerbeli elhelyezkedéséről, a rendszerben található számítógépek típusáról, számáról, stb. Az ilyen rendszerek lehetőséget biztosítanak az egyszerű bővíthetőségre, az erőforrások célszerűbb kihasználására, stb. Az elosztott rendszerek által nyújtott előnyök:

- Egységes file megnevezés (naming scheme)

  A rendszer minden felhasználó számára biztosít egy egységes file-rendszert, amelyben az egyes file-ok ugyanazon módon érhetők el függetlenül attól, hogy fizikailag hol helyezkednek el. (Az egy alkönyvtárban elhelyezkedő file-ok lehetnek akár különböző gépeken is, a felhasználó mégis egy könyvtárban látja őket.)

- A felhasználók szabadon mozoghatnak a terminálok között

  Minthogy a rendszer virtuális file-rendszert biztosít a felhasználó számára, ez független attól, hogy a felhasználó melyik terminálon jelentkezik be.

- File-ok cach-elése

  Annak érdekében, hogy a file-hozzáférést gyorsítsa, a rendszer a más gépeken elhelyezkedő file-okat (vagy azok egy részét) automatikusan a felhasználó lokális háttértárára másol(hat)ja. Ekkor a továbbiakban a file-hozzáférések a helyi háttértáron történnek, gyorsítva ezzel azokat és egyúttal csökkentve a hálózati kommunikációt.

- Nagyobb adatbiztonság a file-ok megtöbbszörözésével

  Az egyes file-ok mozgását (cache-lés, stb.) a rendszer nyilvántartja. Ilyenkor az előző file nem törlődik le automatikusan. Amennyiben a hálózat hibái, vagy egy számítógép kiesése miatt egy file elérhetetlenné válna, az előző verzió továbbra is elérhető, és az adatok (esetleg korlátozott mértékben) továbbra is felhasználhatóak.

- Automatikus újrakonfigurálás

  A rendszer hibák esetén, vagy új számíógépek bekapcsolásakor, hibák elhárulásakor automatikusan újrakonfigurálja magát.

- File-ok elérhetősége jobb

  A file-ok megtöbbszörözése miatt egy számítógép kiesése esetén a file-ok (vagy azok egy része, esetleg régebbi verziói) továbbra is elérhetőek maradnak. Egy gép kiesése nem akadályozza egy másik gépen levő file elérését.

- Egyszerűbben bővíthető

  Mivel a rendszer eleve tartalmaz(hat) automatikus terheléselosztó funkciókat (a virtuális file-rendszer ezt egyszerűen lehetővé teszi), a meglevő eszközök jobban kihasználhatók. Az automatikus újrakonfigurálás lehetősége pedig egyszerűvé teszi új eszközök hozzáadását a rendszerhez.

- Biztonságosabb rendszer

  Mivel a rendszer egyes elemei fizikailag különböző helyeken helyezkednek el, a rendszer nagyobb biztonságot nyújt az illetéktelen hozzáférések ellen.

Az elosztott rendszerek hátrányai:

- Bonyolultabb rendszer

  Annak érdekében, hogy a fent említett előnyöket biztosítani tudja, egy elosztott rendszer lényegesen bonyolultabb lehet, mint a szokásos rendszerek. Ez a bonyolultság azonban megtérül a mindennapi használat során jelentkező előnyök kihasználása során.

Természetesen nem minden elosztott rendszer rendelkezik mindazokkal az előnyökkel, amelyeket fentebb említettünk. Az adott rendszernek minden esetben tekintetbe kell vennie a várható kívánalmakat, az előnyök megvalósításának költségeit, a várható hardware korlátokat, stb.

A következőkben röviden három különböző elosztott rendszert vizsgálunk meg. Mindhárom rendszer más módon próbálja megközelíteni az elosztott rendszerek kérdéskörét, felölelve az ezen rendszerekkel kapcsolatos megoldások széles skáláját.

Network File System

A Network File System (NFS) elnevezés egyszerre jelent egy lokális hálózatokon távoli file-ok elérésére szolgáló szoftver rendszer specifikációt, valamint annak egy megvalósítását (szoftver terméket). Míg a megvalósítás eredetileg az ún. SunOS operációs rendszer (egy UNIX változat) része volt, ma már gyakorlatilag ipari szabvánnyá vált, rengeteg egyedi megvalósítás létezik különbféle számítógépekre.

Egy NFS rendszer egymástól független, hálózaton összekötött számítógépekből áll, amelyek mindegyikének saját, független file rendszere van. Célja az, hogy transzparens módon tegyen lehetővé bizonyos fokú file, ill. file rendszer megosztást az egyes számítógépek között. Ez a megosztás az ún. kliens-szerver architektúrán alapul oly módon, hogy egy számítógép egyszerre láthat -és gyakran lát is- el kliens és szerver funkciókat is. A file, ill. file rendszer megosztás bármely két számítógép között megengedett, nem pedig egy vagy több szerver és a kliensek között.

AMOEBA

Az Amoeba project egy olyan kutatási feladatként indult, melynek célja azon alapelvek kidolgozása volt, amelyek lehetővé teszik több számítógép kényelmes és hatékony összekapcsolását. Az alapelv a felhasználók számára egy egyedi időosztásos rendszer illúziójának megteremtése volt, miközben a valóságban maga a rendszer több, egymással (akár országokon keresztül) összekötött számítógépen "oszlik meg". A cél tehát egy olyan elosztott rendszer volt, amely teljesen "transzparens" a felhasználók felé. A hálózati operációs rendszerekkel ellentétben itt a felhasználók nem egy meghatározott számítógépre jelentkeznek be, hogy onnan azután elérjék a hálózat szolgáltatásait, hanem egy transzparens elosztott rendszerbe. Mikor a felhasználó elindít egy programot, akkor a rendszer -és nem a felhasználó- az "aki" eldönti, hol (melyik gépen) a legcélszerűbb azt futtatni. Sőt a felhasználó nem is tudja a döntés eredményét! Végül, de nem utolsósorban,

az Amoeba rendszer egy egységes file rendszert biztosít a felhasználónak. Az egy alkönyvtárban levő file-ok valójában különböző számítógépeken lehetnek, akár egy másik országban is. Egy file-nak a könytár struktúrában elfoglalt helye nincs összefüggésben annak fizikai helyével.

Plan 9

A Plan 9 rendszer egy általános célú, többfelhasználós, hordozható elosztott rendszer. Mivel parancsai, könyvtárai, és rendszerhívásai hasonlatosak a UNIX rendszeréhez, az átlagos felhasználó csak minimális különbséget észlel a két rendszer között.

Ami a Plan 9 rendszert megkülönbözteti, a felépítése. A rendszer a szolgáltatások mentén osztott. Diszknélküli CPU szerverek koncentrálják a számítási teljesítményt nagy multiprocesszorokba, file szerverek szolgálnak az adatok tárolására, a felhasználók pedig bitmap terminálok előtt ülnek, amelyek egy "windowing" rendszert futtatnak. Mivel a CPU szerverek és a terminálok ugyanazt a kernelt használják (futtatják), a felhasználó szabadon eldöntheti, hogy a programokat saját terminálján, vagy a távoli CPU szerveren óhajtja futtatni. Így azután a felhasználók, ill. rendszer adminisztrátorok olyan elosztottra, ill. centralizáltra konfigurálhatják rendszerüket, amilyenre óhajtják.

Bibliográfia

Dave Presotto, Rob Pike, Ken Thompson, Howard Trickey: **Plan 9, A distributed system**
EurOpen Conference Proceedings, Spring 1991

Andrew S. Tanenbaum, Robbert Van Renesse, Hans Van Staveren, Gregory J. Sharp, Sape J. Mullender, Jack Jansen, Guido Van Rossum: **Experiences with the AMOEBA distributed operating system**
Communications of the ACM, December 1990/Vol 33. No.12/

Eliezer Levy, Abraham Silberschatz: **Distributed file systems: Concepts and Examples**

ACM Computing Surveys, Vol.22, No.4, December 1990

A szerver-kliens architektúra alkalmazása Magyarországon

Sziebig Ferenc - Microsystem

Egy sokfelhasználós alkalmazás legkorszerűbb működési módja napjainkban a szerver-kliens architektúra. Ez a megoldás a centralizált feldolgozással szemben kihasználja a hálózatba kötött számítógépek osztott intelligenciáját, viszont az alkalmazás különböző fizikai helyeken futó részei között a **kapcsolat logikai jellegű**, kevésbé terhelve a hálózatot és a rendszer összteljesítményét jelentősen növelve. A kapcsolat jellegéből adódóan ez a hálózat nem kötelezően csak lokális lehet, a szerver-kliens architektúra elemeit összeköthetjük **WAN hálózat is.** Piaci előrejelzések szerint a következő öt évben az osztott on-line információs rendszerek területén ez a szegmens fog a legdinamikusabban fejlődni, így ez a működési elv lesz domináns multiuser-es környezetben.

Előadásomban - a Microsystem, a **Data General** magyarországi disztribútora képviseletében - ezen az új és perspektívikus területen elért eredményeinkről szeretnék beszámolni, valamint szólni a további terveinkről.

Egyik érdekes alkalmazása a szerver-kliens architektúrának a balatonfűzfői **Nitrokémia Ipartelepek** termelésirányítási rendszerének megoldása, amelynek logikai sémája az 1. ábrán látható. A **Progress** alapú termelésirányítási alkalmazás két duálprocesszoros gépen fut, az egyik gép az adatbázis szerver, a másik gépen fut az összes kliens program. Megkérdezheti valaki: miért nem alkalmaztunk inkább egy négyprocesszoros típust? Ezzel a konfigurációval az igazi hibatűrő rendszerekhez képest sokkal alacsonyabb áron szinte egy teljes hibatűrő megoldást sikerült elérnünk. A két példányban írt adatbázis egyik példánya egy külső diszken található, amelyet az **egyik gép kiesése esetén elérhet a másik**. Így a teljes rendszer összteljesítménye megegyezik egy négyprocesszoros gépével, s rádásul a Progress szerver-kliens működésének következtében az alkalmazás ezt egy gépként kezelheti. Az egyik gép leállása esetén a teljes rendszer - fele teljesítménnyel ugyan, de továbbra is hozzáférhető.

Ide kívánkozik egy gyakran felmerülő feladat: adott **egy WAN hálózat**, sok esetben X.25, vagy bérelt vonal. Ezen hálózat több fizikai telephelyet köt össze, melyeken egy adott **egységes felhasználói rendszer** bizonyos részei futnak. Ezek a részek a legtöbb esetben lokális adatokkal helyben dolgoznak, viszont néha **adatokra van szükségük más színhelyekről**, vagy adatokat szolgáltatnak egy központi információs rendszer részére. Tudjuk, hogy nagy adatbáziskezelők nyújtanak teljes és transzparens megoldást erre a problémára

(osztott adatbáziskezelés, STAR modulok), de ezek az eszközök a legtöbb esetben túl nagynak és drágának bizonyulnak. Figyelembe véve a vonalak lassúságát, a legoptimálisabb megoldás ebben az esetben, ha valamely szerver-kliens üzemmódban működni tudó adatbáziskezelővel dolgozunk, s amikor az alkalmazás egy másik színhelyet kell hogy elérjen, az applikációs program **a másik csomópont szerverével is felveszi a kapcsolatot** (általában a NET réteg segítségével). Mivel a kommunikáció SQL nyelven történik (tehát logikai szinten), ebben az esetben függünk a legkevésbé a vonali sebességektől. Egy jól használható eszköz az előzőek kivitelezésére pl. az **ORACLE** adatbáziskezelő. Ezt az architektúrát mutatja a 2. ábra.

Egy másik érdekes gyakorlati alkalmazása a szerver-kliens architektúrának az **Alkaloida Vegyészeti Gyár** új kiszerelő üzemének folyamat-irányítási, termeléskövetési rendszere. Ebben az esetben a lokális munkaállomások helyben végzik a folyamatirányítási tevékenységet (kapcsolat mérlegekkel, PLC-kkel), de tevékenységükhöz már egy központi adatbázisból kapnak információkat, illetve az egész folyamat követéséhez folyamatosan szolgáltatnak adatokat egy központi adatbázis számára. A fenti folyamat a gyakorlatban úgy valósul meg, hogy a munkaállomásokon a **Progress kliens oldala fut a folyamatirányítási modulokkal integráltan**, s egy lokális hálózaton keresztül vannak állandó kapcsolatban a központi adatbázis szerverrel.

Szintén gyakori a következő eset: a felhasználó tovább szeretne lépni, a meglévő Novell-Clipper alapú felhasználói rendszerét **egy megbízhatóbb minigépes megoldással felcserélve**. Az áttérés mindig felveti a következő kérdéseket:

- a meglévő PC-k az új rendszerben is használhatók legyenek,
- az áttérés folyamatos és zökkenőmentes legyen,
- lehetőleg az ismert felhasználói rendszer kerüljön át az új platformra.

A fenti követelmények mind kielégíthetők a következő módon:

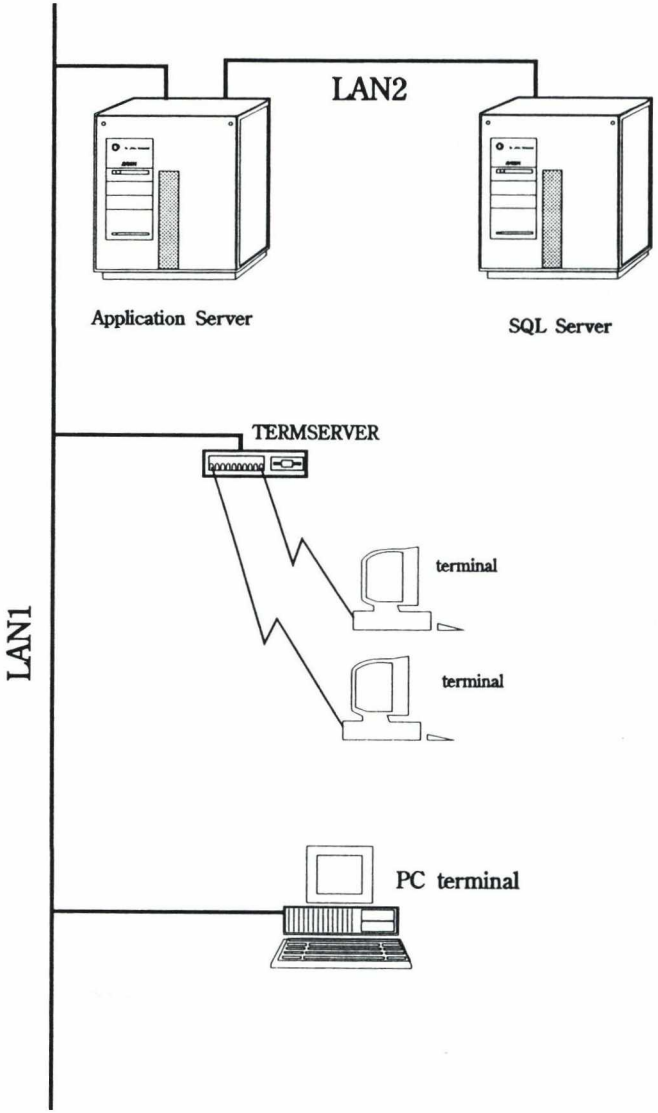
A meglévő hálózatba (feltéve persze, ha Ethernet) betelepítjük a minigépet is. Ezután a meglévő PC-k minőségüktől függően érhetik el az új gépen futó új rendszert:

- ha 286-os, vagy még kisebb processzorral és kevés memóriával rendelkeznek, mint PC terminálok használhatók,
- ha 386SX, vagy erősebb processzort tartalmaznak, illetve van bennük bővített memória és lokális fix diszk, futhat rajtuk az alkalmazás kliens oldala,
- ezek a típusok persze keverhetők azonos rendszeren belül is.

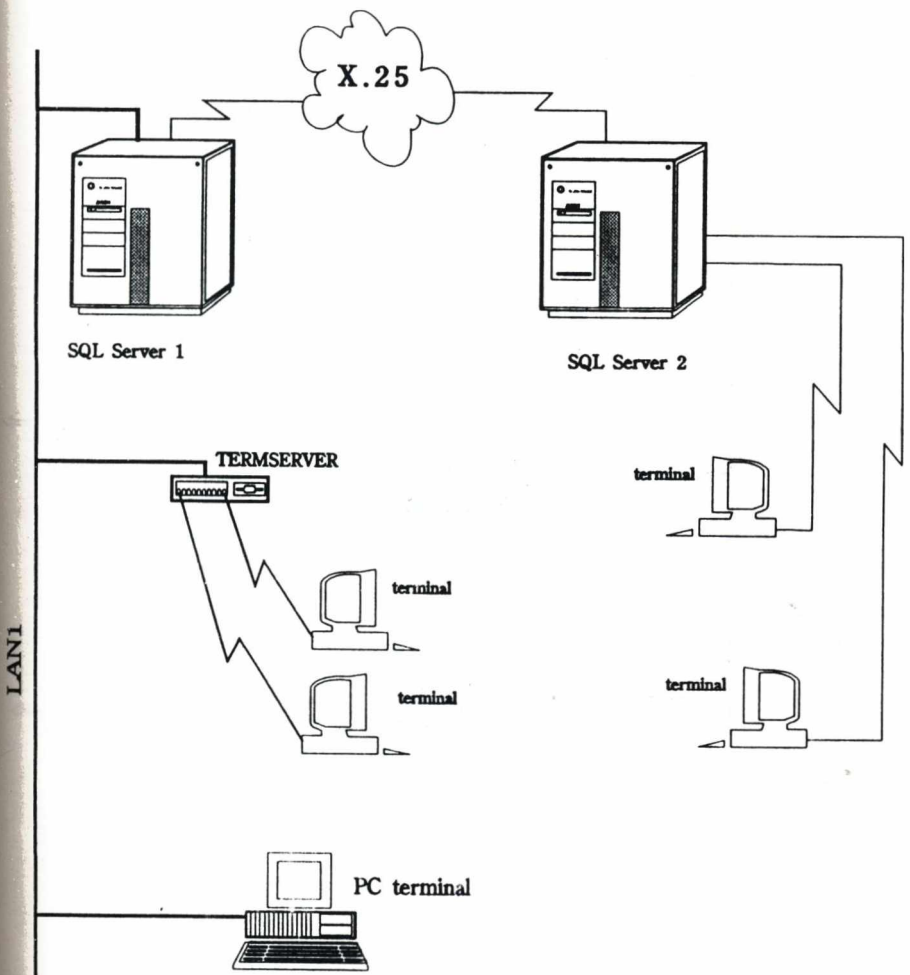
Megoldható az is, hogy azonos munkahelyen a régi és az új alkalmazás felváltva fusson. Ezt a konfigurációt láthatjuk a 3. ábrán.
A **TOPSOFT** Kft. **AViiON** gépet használva, a fent ismertetett környezetben óhajtja azt a kívánságot is kielégíteni hogy az ismerős felhasználói rendszer fusson az új gépen korszerű eszközökkel megvalósítva: a sikeres integrált vállalati ügyviteli programcsomagját írja újra **INGRES**-ben.

Végezetül egy, a közeljövőben már elérhető, a szerver-kliens architekturán alapuló **komplex rendszer** képét szeretném vázolni a 4.ábrán. Ebben az esetben a hálózatban már nem csak adatbázis szerverek érhetőek el hanem - a logikai szerver fogalmának általánosításaként - más funkciókat ellátó szerverek (kommunikációs, fax, stb) is.
A szabvány grafikus felhasználói felülettel (pl. MS-Windows) rendelkező kliens munkaállomásokon párhuzamosan futhat, (külön-külön ablakban) egy adott 4GL alkalmazás (amely adott esetben akár többféle adatbázis szerverrel is együtt dolgozhat), valamint egy **elektronikus irodai programcsomag és dokumentum archiváló rendszer** kliens oldala, amely munkája közben igénybe veszi a hálózatban található image, fax és egyéb szerverek szolgáltatásait.

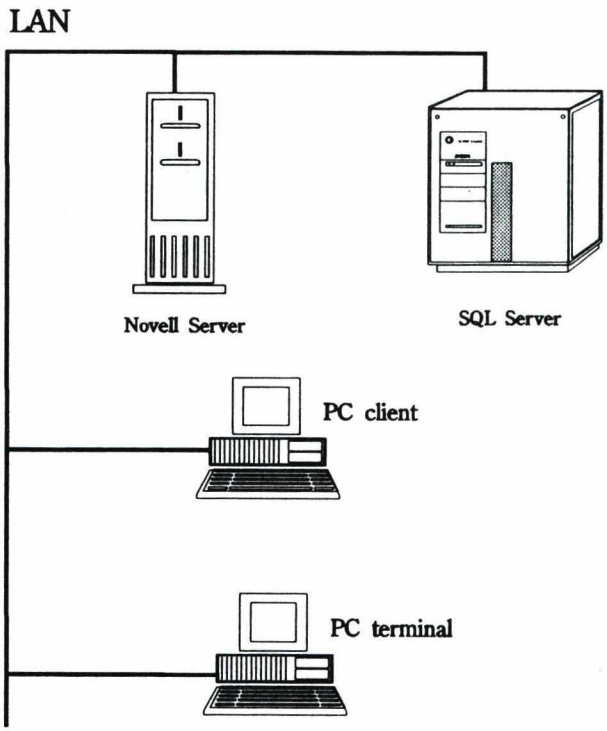
1. ábra


LAN2
Application Server
SQL Server
TERMSERVER
terminal
terminal
LAN1
PC terminal

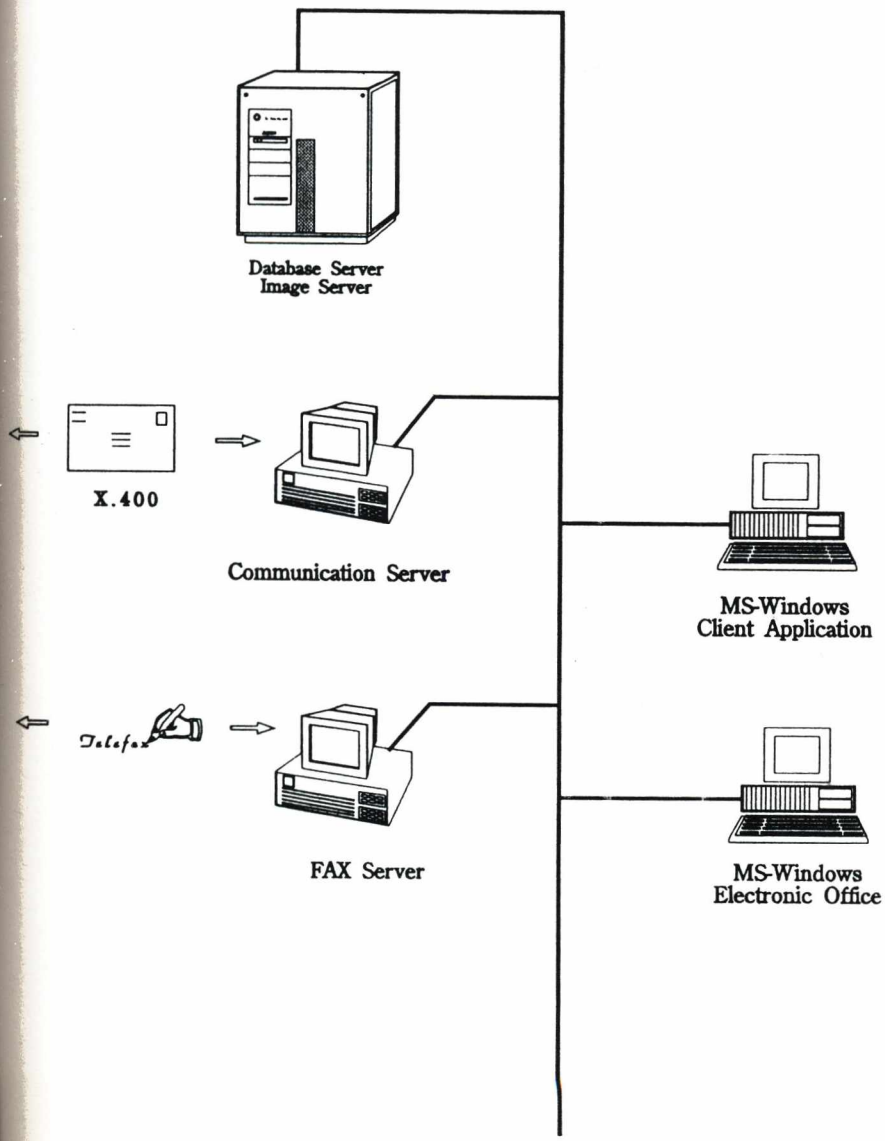
2. ábra


X.25
SQL Server 1
SQL Server 2
TERMSERVER
terminal
terminal
terminal
terminal
PC terminal
LAN1

3. ábra



4. ábra


Database Server
Image Server
X.400
Communication Server
MS-Windows
Client Application
Telefax
FAX Server
MS-Windows
Electronic Office

IV.szekció: **ADATBÁZIS KEZELÉS U N I X KÖRNYZETBEN**

RECITAL

negyedik generációs adatbázis-kezelő rendszer

Tuba Zoltán, ISYS Számítástechnikai Kft.

Bár Magyarországon ma még csak kevéssé ismert, több szempontból egyedülálló a 4GL piacon a hatékony fejlesztő eszközöket és funkcionálisan sokoldalú interfész modulokat tökéletesen integrált környezetben magába foglaló szoftvertermék: a Recital.

A Recital negyedik generációs alkalmazásfejlesztő környezet (4GE) és relációs adatbázis-kezelő rendszer az információtechnológia ezen ágának mintegy másfél évtizedes fejlődése során kialakult szabványos elemekre épül, olyan fejlesztői és végfelhasználói környezetbe integrálva azokat, amelyek segítségével

- a fejlesztés ideje lényegesen lerövidíthető,
- az egymástól eddig elkülönült információ források egységesen kezelhetők,

mindezt a hagyományos relációs megoldásoknál lényegesen kedvezőbb áron.

Mi ebben a különös eddig, hiszen az úgynevezett 4GL termékek széles választékát kínálja ma a piac. Mik azok a tulajdonságok, amelyek megkülönbözetik a Recital-t a rajta kívül létező mintegy 167 másik 4GL-től?

A legfontosabb dolog, ami a Recital-t hasonló társaitól megkülönbözteti, az alkalmazásfejlesztés szemléletbeli megközelítése. A konkurrens termékek legnagyobb része arra kényszeríti az alkalmazót, hogy meglévő tudását, tapasztalatait sutba dobva tanuljon bele egy vadonatúj fejlesztő környezetbe. A Recital esetében erről szó sincs. Hogy miként lehetséges ez? Lássuk csak ...

A Recital 4GL/RDBMS bevezetése egy szervezet keretein belül az alábbi előnyökkel jár:

- A meglévő szaktudás továbbra is használható.
- **A meglévő dBASE, FoxBase, és Clipper alapú alkalmazások csekély erőfeszítéssel és kis költséggel átültethetők az új rendszerbe,** hardver platformok és operációs rendszerek széles választékát nyitva meg ezáltal a szóban forgó alkalmazás számára.
- Az alkalmazásfejlesztők már az első naptól kezdve produktívak lehetnek. Nincs szükség hosszadalmas és drága átképzési időszakra, vagy költséges konzultációs szolgáltatások igénybevételére.
- Olyan fejlesztői környezetet biztosít, amely UNIX és VMS alapú rendszerek alatt is lehetővé teszi "PC-szerű" alkalmazások kidolgozását. A PC-s környezetben megszokott képernyőelemek (popup és pulldown menük stb.) akár az egyeszerű terminálokon is megjeleníthetők.
- Lehetővé teszi az alkalmazások objektum-orientált szemlélettel történő fejlesztését.
- Kényelmes fejlesztői eszközei segítségével az alkalmazások jelentékeny része programozási művelet nélkül kidolgozható. A fejlesztő eszközök használata nem csökkenti a programnyelv kínálta funkcionális lehetőségeket.
- Nemcsak a saját, többszintű védelemmel és hatékony adatszótárral ellátott adatbázisait tudja kezelni, hanem transzparens hozzáférést biztosít más, elterjedt adatbázis-kezelőkkel létrehozott adatbázisokhoz is.
- Nyílt, kliens-szerver architektúrája révén a saját vagy idegen adatbázisokhoz való hozzáférést elosztott rendszereken, hálózati kapcsolatokon keresztül is lehetővé teszi.

A Recital valamennyi elterjedt UNIX-os rendszeren (több, mint 100 platform), valamint a DEC VMS és ULTRIX alapú gépein fut. A Recital alatt kidolgozott alkalmazások valamennyi támogatott platformon módosítás nélkül futnak.

1. A Recital koncepciója: megszakítás nélküli átmenet

A hardver és szoftver rendszerek utóbbi években bekövetkezett ugrásszerű fejlődése előtérbe hozta az inkompatibilitási problémákat és az alkalmazások újrafejlesztésének kérdését.

A Recital egyedülálló koncepciója miatt nem igényli a meglévő rendszerek stratégiai átalakítását, nem küszködik kompatibilitási problémákkal, lehetővé teszi az eddig elkülönülten létező adatállományok egységes rendszerben történő kezelését, biztosítja az alkalmazás-fejlesztésre fordított invesztíciók és a fejlesztés során létrejött szakértelem védelmét.

Nagymértékben csökken az alkalmazások újbóli kidolgozásakor óhatatlanul felmerülő sokféle hibalehetőség, amely az új rendszer nem eléggé alapos ismeretéből, a gyakorlat hiányából fakad. Nincs szükség az alkalmazások újratervezésére és újrakódolására, az információs bázis pedig a meglévő adatokból építhető fel.

Nem jelent problémát az újabb, a technológia mindenkori fejlettségi szintjének megfelelő hardver és operációs rendszer fejlesztések kínálta előnyök kiaknázása sem.

2. A Recital mint negyedik generációs alkalmazásfejlesztő környezet

A Recital talán legvonzóbb tulajdonsága az alkalmazásfejlesztés hatékonysága. A Recital alatt nagyságrenddel rövidebb idő alatt dolgozhatók ki alkalmazások, mint más adatbáziskezelő rendszerekben.

Becslések szerint az alkalmazásfejlesztésre fordított idő átlagosan 50 %-át tölti ki a felhasználói felületek (user interface) megtervezése és az adatok integritásának biztosítása. A Recital minimálisra csökkenti ezt az időtartamot. Az adatok integritásáért egy központi, mindig aktív adatszótár (Applications Data Dictionary) a felelős. Az üzletorientált alkalamazások számára tervezett, beépített elemkészlet segítségével pedig az egységes képernyők felépítésének folyamata automatizálható. Ugyanebből az elemkészletből építkeznek a Recital saját eszközei és felhasználói felületei is.

Ha megvizsgálunk néhány másik 4GL-t is ugyanebből a szempontból, azt tapasztaljuk, hogy az alkalmazások kidolgozására szolgáló eszközök funkcionalitásban és tulajdonságaikban is eltérnek a velük fejlesztett alkalmazásoktól. A fejlesztő eszközök többsége magából az alkalmazásból nem hozzáférhető. Mindez hosszabb betanulási időt és fejlesztési folyamatot eredményez.

A Recital segítségével az alkalmazások rövid idő alatt kifejleszthetők, különösebb betanulási idő nélkül. A kész alkalmazások könnyebben használhatók, karbantartásuk egyszerűbb.

2.1. A Recital/Assistant

A Recital/Assistant PC-szerű felhasználói keretet biztosít ahhoz, hogy a Recital akár a fejlesztő, akár a végfelhasználó számára egyszerűen használható legyen, akár különösebb gyakorlat nélkül is. A PC-s eszközökből már ismerős képernyőelemek, pop-up és ring-stílusú menük, a táblázatkezelők (pl. Lotus) mintájára működő különféle szolgáltatások olyan, barátságos környezetet teremtenek, amelyben biztonsággal mozoghat a PC-s gyakorlattal rendelkező felhasználó. A munka valamennyi fázisában on-line help áll rendlekezésünkre.

Az Assistant, amely maga is Recital/4GL nyelven íródott, élő példája annak, hogy milyen színvonalú alkalmazások fejleszthetők ki, minimális programkódolással.

2.2. Az interaktív parancs üzemmód

Gyakorlott alkalmazásfejlesztők a menüvezérelt Assistant helyett interaktív parancs üzemmódban is dolgozhatnak. A parancs üzemmódból a Recital valamennyi fejlesztő eszköze és szolgáltatása hozzáférhető.

Az adatbázisok interaktív módon történő kezelésével sem sérülhet meg az adatok integritása, mivel az teljes mértékben az adatszótár felügyelete alatt van. Elkerülhető ezáltal a kényes adatokhoz történő illetéktelen hozzáférés a fejlesztés és a karbantartás során is.

A dBASE (FoxBASE, Clipper) környezetben otthonosan mozgó programozók ismerősnek fogják találni a Recital interaktív parancs üzemmódját. Természetesen a Recital szolgáltatásai messze meghaladják a dBASE és dialektusainak lehetőségeit.

3. A Recital negyedik generációs nyelv

Kiemelkedően sokoldalú és rugalmas negyedik generációs nyelv a Recital/4GL. Az a lehetőség, hogy a megfelelő elemkészlet kiválasztásával építhetünk fel teljes alkalmazásokat, rövid fejlesztési ciklusidőt biztosít.

A Recital alatt kifejlesztett alkalmazások figyelemreméltóan szép kivitelű és megbízhatóan működő felhasználói interfésszel rendelkeznek. A pop-up menük, választéklisták stb. automatikusan tárják a felhasználó elé a szükséges információt, a háttérben állandóan jelenlévő adatszótár révén.

Számos olyan szolgáltatás, amely más 4GL környezetben kódolási műveletet igényel, a Recital alatt beépített elem. A Recital alatti fejlesztések során a fő hangsúly a tervezésen van, nem pedig a kódoláson.

3.1. A dBASE nyelvcsalád

A Recital felülről kompatibilis az összes népszerű dBASE dialektussal: a dBASE III PLUS-szal, a Clipper summer'87-tel, a FoxBASE v2.1-gyel és a dBASE IV-gyel, így biztosítható az e nyelveken megírt alkalmazások átültetése (portolása) és a megszerzett tudás továbbvitele, ami a Recital két legfőbb célja.

Több tízezerre tehető a dBASE dialektusokban megírt alkalmazások száma világszerte. A Recital segítségével ezek mindegyike több, mint 100 támogatott platformon válik futtathatóvá.

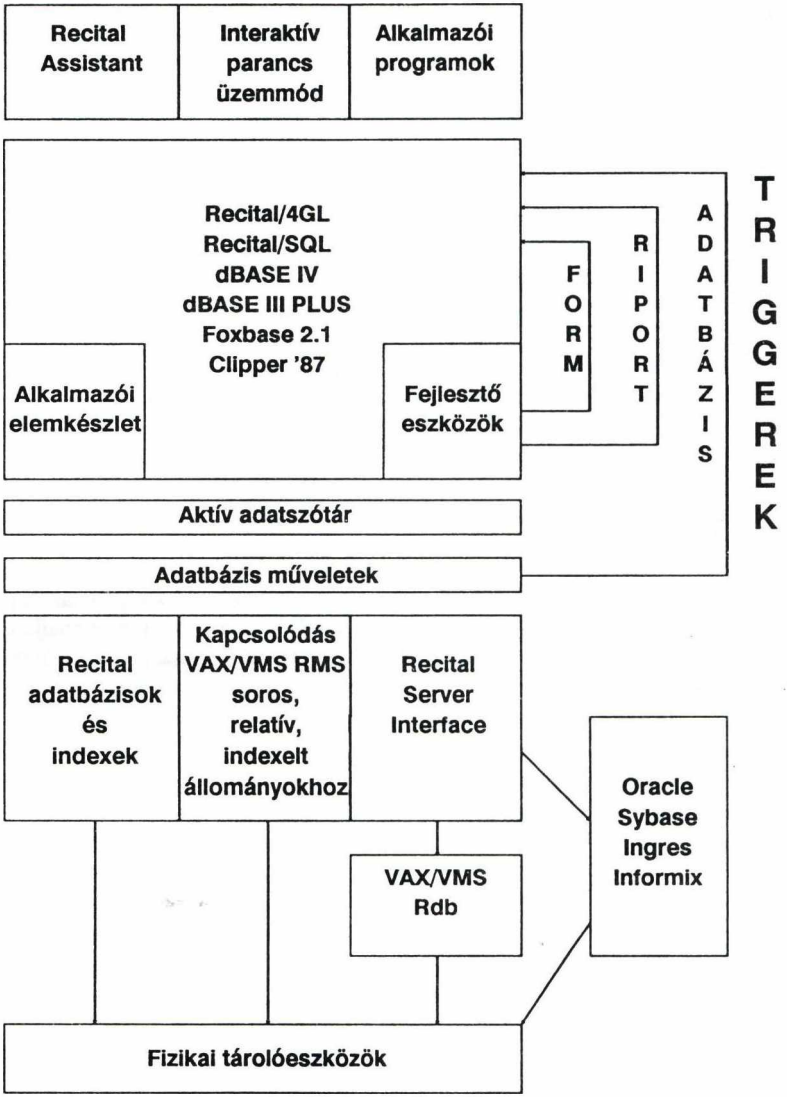
3.2. A Recital/SQL

A beépített függvények és az alkalmazási elemkészlet segítségével sok kódolási feladat egysoros utasítássá válik. A menürendszerből elérhető szolgáltatásokat kihasználva a felhasználónak igazából nincs is szüksége arra, hogy megtanuljon valamilyen lekérdező nyelvet. Mégis, a Recital mint a nyílt rendszerarchitektúrák és a szabványok elkötelezettje az ANSI SQL-t is integrálja magában, amelyet interaktív parancs üzemmódban vagy programba ágyazva egyaránt használhatunk.

4. Transzparens kapcsolat SQL adatbázisokhoz

A Recital nyitott rendszerarchitektúrája a Recital Server Interface segítségével transzparens hozzáférést biztosít az ipari szabványoknak megfelelő adatbázisokhoz. Az adatbázisok olvasásra és írásra egyaránt megnyithatók, és összekapcsolhatók akár a Recital saját, akár más egyedi formátumú adatbázisokkal. A támogatott adatbázisok: Oracle, Informix, Ingres, Sybase, valamint a VAX/Rdb.

A Recital architektúrája



SQL ALKALMAZÁSOK PORTABILITÁSA EGY KONKRÉT PROJEKT ALAPJÁN

Gacsai Gábor
Műszaki Software-Fejlesztő Kft.

A UNIX környezetben elterjedt SQL alapú relációs adatbáziskezelők (INFORMIX, INGRES, ORACLE DDB4 stb.) felépitésük szempontjából jelentős eltéréseket mutatnak, ugyanakkor funkcionalitásukat tekintve, egy többé-kevésbé hasonló felhasználói interfészt biztositanak:

- Interaktiv monitort, amely SQL parancsok közvetlen végrehajtásának eszköze;
- Képernyőkezelést támogató maszk vagy form generáló eszközöket, amelyek a karbantartó funkciók mellett egyszerűbb lekérdező/kereső szolgáltatást is nyújtanak;
- Report generátort;
- Negyedik generációs (4GL) eszközöket
- 3. generációs nyelvekhez (C, COBOL, FORTRAN ...) készült SQL interfészt, amit egy precompiler segitségével alakithatunk át pl. szokásos C programmá.

Sajnos a hasonlóság csak funkcionális, az eszközök szintaxisának és szemantikájának különbözőségéből adódóan az egyik adatbáziskezelőre irt alkalmazás általában még forráskódi szinten sem vihető át a másik adatbáziskezelőre. A kivétel(-nek látszik) az az eset, amikor az alkalmazást 3. generációs - tehát már szabványositott - nyelven irjuk meg, és az adatbáziskezelők felületei közül a beágyazott SQL - tehát szintén kvázi-szabványos - parancsokat használjuk.

A szerző részt vett egy UNIX alatt futó, C-be ágyazott SQL hivások segitségével megvalósitott alkalmazásnak különböző adatbáziskezelő környezetekbe történő "átültetésében". Az előadásban a teljesség igénye nélkül elsősorban azokra a buktatókra hivja fel a figyelmet, azon tapasztalatokat ismerteti, amelyek ezen portálás során keletkeztek.

A portálandó szoftver - egy komplett termelésirányitási rendszer - eredetileg NIXDORF gépen a DDB/4 adatbáziskezelőre irodott, C-be ágyazott SQL segitségével, felhasználva a DDB/4 4. generációs fejlesztőeszközét a DIALOG-ot is. A DIALOG ugyan gyorsabb programfejlesztést tett lehetővé, viszont az igy megirt programok, esetenként lényegesen lassabban futottak. A későbbiek során a DIALOG

programok ellen még egy nagyon fontos érv szólt: más adatbáziskezelői környezetre nem voltak átvihetők. Emiatt az újabb hardver és UNIX környezetekben (TARGON, DEC és SCO) az alkalmazás már csak a DDB/4 C-be ágyazott SQL interfészét használta. A vevőkör további bővülése magával hozta az adatbáziskezelői környezet megváltoztatásának igényét is.
A programrendszert úgy kellett módositani, hogy az HP és DEC gépeken INFORMIX alatt is futóképes legyen, figyelembe véve olyan egyéb szempontokat is, amelyek a későbbiek során akár más relációs adatbáziskezelőkre (elsősorban INGRES-re és ORACLE-re) is lehetővé teszik az portálást.
(Az előadás folyamán ezen DBMS-ek alatt azok 1990 végén élő verzióit értjük, tehát DDB/4 2.4, INFORMIX 4.0, INGRES 6.2, ORACLE 6.0.)

Az alkalmazás hordozhatósága az SQL szintjén a C-be ágyazott SQL használatával elméletileg biztositottnak látszott, igy a portálás az alábbi főbb témakörökre volt bontható:

- A rendszeradminisztrátori szinten jelentkező DBMS-sajátosságok felderitése, az egyes adatbáziskörnyezetek létrehozása, valamint egyéb globális jellegü feladatok elvégzése;

- A beágyazott SQL interfész adatbázisfüggő részeinek meghatározása mind az SQL mind pedig a precompiler szintjén, majd pedig ezen DBMS-függő részek lokalizálása, ill. lehetőség szerinti megszüntetése az alkalmazásban.

I. Adminisztrációs jellegü és egyéb globális feladatok

1. Az adatbáziskörnyezet létrehozása

A programcsomag müködéséhez egy bizonyos adatbáziskörnyezet szükséges. Ez magában foglalja egy adatbázis és a hozzá tartozó objektumok / táblák, indexek, view-k ... /, valamint az esetleg szükséges adatbázis-felhasználók meglétét is.
Az erre szolgáló parancsok szintaxisa és szemantikája is esetenkét jelentősen eltérhet, igy erre különös figyelmet kell forditani. Ilyen parancsok a:

```
CREATE DATABASE ...
CREATE TABLE ...
CREATE INDEX ...
```

Az adatok áttöltéséhez az egyik DBMS-ből a másikba az egyes DBMS-ek adatbáziskimentő és betöltő segédprogramjait használtuk fel. Ilyennel kivétel nélkül mindegyik rendelkezik, sajnos azonban ezek input és output formátumai nem mindig felelnek meg egyértelmüen egymásnak, igy itt szükség volt kisebb konvertáló programok használatára.

Ezután következnek az egyéb DBMS-függő adminisztrációs feladatok: az esetleges felhasználók, ill. felhasználói-csoportok definiálása, a tulajdonosi és hozzáférési jogok

beállitása. Némely adatbázishoz külön kell adatbázis-felhasználókat és csoportokat definiálni és ezekhez password-ot rendelni, másutt csak a már meglevő UNIX felhasználóknak kell a megfelelő hozzáférési privilégiumokat kiosztani. Ebből következik, hogy az e célt szolgáló:

```
CREATE USER ...
GRANT ...
```

parancsok szintén DBMS specifikusak.

Adminisztrátori szinten kellett a public, ill. privát táblák elérésének problémáját is megoldani:
DDB/4-ben az un. public táblákat mindenki elérheti, a privát táblákat pedig egy adott felhasználó vagy felhasználói csoport számára definiálhatjuk. Az azonos nevű, de különböző tulajdonosú táblák megkülönböztetésére a "tulajdonos.táblanév" megnevezést használhatjuk. Ha egy táblára tulajdonosának megjelölése nélkül hivatkozunk, akkor a DBMS először megnézi, hogy az aktuális felhasználóhoz tartozik-e ilyen nevű privát tábla, és csak ha ilyen nincs, akkor nyúl a "public.táblanév" táblához.
Az eredeti alkalmazásban a táblákra mindig minősités nélkül, azaz csak "táblanév"-ként történik hivatkozás. Ily módon lehetett a különböző verziójú táblákat az alkalmazásban egységesen kezelni. Ha valamelyik felhasználói csoportnak egy új verziójú táblára volt szüksége, akkor létrehozta a csoporton belül az új táblát, de az alkalmazásban nem kellett sem az adatbázis-, sem a tábla-hivatkozásokat módositani.

Mivel ez a tulajdonság egyéb DBMS-ekben funkcionálisan másképp valósitható meg, ezért itt adatbáziskezelőnként különböző megoldások születhetnek.
INFORMIX-nál például a SYNONYM-ot használtuk e probléma megoldására.

2. A DBMS és az alkalmazás finomitása a hatékonyság érdekében (tuning)

Az adatok fizikai elhelyezkedése a táblákon, illetve az adatbázison belül az alkalmazás hordozhatóságának szempontjából ugyan nem játszik jelentős szerepet, annál inkább érinti viszont annak hatékonyságát, igy az ezzel kapcsolatos ismeretek fontos szerepet játszhatnak már az implementálás kezdetén is.

Az adatbázis és a táblák tárolási struktúrájának kialakitása, az elsődleges és másodlagos indexek valamint a kulcsok létrehozása - a DBMS-ek eltérő felépitése miatt - az a terület, ami a szabványositási törekvések ellenére még mindig nagy eltéréseket mutat.

Az egyes rekordok gyors elérését a különböző indexek, vagy esetleg a tábla tárolási struktúrája biztositja.

Az indexek létrehozására a "CREATE INDEX" parancs szolgál, amely adatbáziskezelönként különböző kiterjesztésekkel rendelkezik.

A táblák tárolási struktúráját különböző módon befolyásolhatjuk:
Van ahol implicit módon - például kulcsok megadásával a "CREATE TABLE parancsban" - ilyen például a DDB4 és az ORACLE. Van ahol a "CREATE INDEX" parancsban a CLUSTER kulcsszó megadásával / INFORMIX, ORACLE /.
Van ahol / INGRES / a rendszeradminisztrátor explicit módon, külön e célt szolgáló paranccsal - "MODIFY táblanév TO struct ON mezo" - szervezheti a táblát az alábbi négy tárolási osztály valamelyikébe: HEAP, HASH, ISAM, BTREE.

Egy adott SQL utasitás (például "SELECT") optimális végrehajtása azonban nemcsak a táblák felépitésétől függ, hanem bizonyos DBMS-eknél függhet a táblák elöfordulási sorrendjétől is a SELECT utasitáson belül. Ez az adott DBMS-nek az adatok eléréséhez használt optimalizálási stratégiájától függ. A fenti jellegü tulajdonsággal bir például a DDB4 és az ORACLE, mig ettöl már független az INFORMIX és az INGRES. Igy fordulhatott elö például a portálás folyamán, hogy egy SELECT utasitás végrehajtási ideje két különböző DBMS környezetben nagyságrenddel tért el egymástól, és bár szintaktikailag minden rendben volt az utasitást mégis illeszteni kellett az adott DBMS-hez.

3. A LOCK kezelés különbözöségei

Az egyes DBMS-ek LOCK kezelése szintén az az ingoványos terület ami a szabványositási kisérletek ellenére még meglehetösen vegyes képet mutat. A gondok a következök:

a. Eltérö finomságú szabályozható LOCK
   /sorszintü, lapszintü, táblaszintü, adatbázisszintü /
b. Az egymással konkurráló processzek esetén egy adott eröforrás olvashatóságának szabályozhatósága (processz izoláció)

Ezen eltérések egyenes következménye a LOCK módot beállitó parancsok különböző szintaxisa és szemantikája. A LOCK móddal kapcsolatos feladatok egy része adatbázis adminisztrátori feladat más része az alkalmazói programok szintjén jelentkezik.

a. Az alkalmazás kihasználta, hogy DDB/4-ben a default LOCK sorszintü. Ez igy történik INFORMIX SE-ben és ORACLE-ben is alapértelmezéses módon. INFORMIX-OnLine-ban a "CREATE TABLE ..." parancsban adható meg, hogy a LOCK sorszintü legyen. INGRES-nél azonban mindenképpen a lapszintü LOCK a legkisebb egység. Ez jelentösen ronthatja az alkalmazás konkurrenciáját, hiszen egy rekord lefoglalása esetén a lapon levö összes többi rekord is lefoglalt állapotba kerül. Az alkalmazásban ezt a tulajdonságot mindenképpen figyelembe kell venni.

b. A LOCK tipusai alapvetően a következőek lehetnek:

- EXCLUSIVE vagy
  WRITE LOCK : egy adott időben egy adott erőforrást csak egy felhasználó foglalhat ily módon le. Más LOCK erre az erőforrásra nem adható.

- SHARED vagy
  READ LOCK : egy adott erőforrást több felhasználó is lefoglalhat igy egyszerre.

Mindkét LOCK tipusra igaz, hogy közben a lefoglalt erőforrást más nem módosithatja.
Más a helyzet azonban az olvasásnál. Itt a processz izolációval a konkurrencia és a konzisztencia egymással ellenkező irányba mozgó szintjeit határozhatjuk meg.

DIRTY READ: nincs LOCK, a processz minden rekordot olvashat, még egy befejezetlen tranzakción belül módositott "fantom" rekordot is.

COMMITTED READ: csak azok a rekordok olvashatók, amelyek SHARED LOCK tipussal lefoglalhatók lennének, tehát befejezetlen tranzakción belül módositott rekord már nem olvasható.

REPEATABLE READ: mint COMMITTED READ, de a beolvasott rekordok SHARED LOCK móddal ténylegesen lefoglalásra kerülnek, igy más már nem módosithatja őket.

Az egyes DBMS-ekben a fenti módoknak csak általában egy részhalmazát lehet megvalósitani, ezért különösen fontos volt olyan szint használata, amely a legtöbb DBMS-ben megvan.
Az alkalmazás úgy készült, hogy a legnagyobb prioritást a konkurrencia kapta, azaz a DIRTY READ LOCK módot használta. Ennek beállitására DBMS-enként különböző parancsok szolgálnak:

```
DDB/4     -   "CONNECT ... LOCK MODE EXPLICIT";
INFORMIX  -   "SET ISOLATION TO DIRTY READ ";
INGRES    -   "SET LOCK MODE ... READLOCK=NOLOCK".
```

További eltérés az egyes DBMS-ek között, hogy mi történik akkor ha egy lefoglalt rekordhoz szeretnénk hozzáférni:

1. nincs várakozás
2. adott ideig történik várakozás
3. "végtelen" ideig tartó várakozás

Ezen paraméterek beállitása a DDB/4-nél részben adminisztrációs feladat, másutt viszont az alkalmazásban kell a megfelelő formátumú SQL parancsokat kiadni.

A LOCK időtartama mindig az adott tranzakció végéig tart. Vannak azonban már olyan SQL kiterjesztések, amelyekben egy LOCK a tranzakción belül is megszüntethető, illetve azon túl is megtartható. Ezen parancsok használatától óvakodjunk, hiszen az ilyen alkalmazások erösen DBMS-függőek lesznek.

Fontos adminisztrátori kérdés a LOCK eszkaláció is, azaz ha a lap- vagy rekordszintü LOCK mennyisége elér egy bizonyos határt, akkor abból tábla szintü LOCK lesz-e, vagy hibaüzenetet kapunk arról, hogy a LOCK tábla betelt. Ez utóbbi esetben a rendszeradminisztrátor feladata az alkalmazás ismeretében a DBMS-t úgy konfigurálni, hogy ez az eset ne forduljon elö. INGRES-nél van LOCK eszkaláció, DDB/4-nél, INFORMIX-nél ORACLE-nél nincs.

A fentiekből következik, hogy ahol csak mód van rá ott az explicit LOCK parancsok használata kerülendő, ha mégis feltétlenül szükséges akkor inkább logikai LOCK-ot használjunk.

II. A beágyazott SQL hivásokat tartalmazó C programok szintjén jelentkező problémák

1. Egyéb eltérő szintaktikájú, ill. szemantikájú SQL parancsok

Az eddig ismertetteken kivül számos egyéb olyan fontos SQL parancs van, amelyiknek ugyan adatbáziskezelönként hasonló funkciója van, mégis más-más a parancs szintaktikája. Ügyelni kell például a minden alkalmazásban meglevő "CONNECT" és "DISCONNECT" jellegü parancsokra. Célszerü az ilyen kénytelen-kelletlen megmaradó DBMS függőségeket az alkalmazásben függvényként elrejteni, és egy könyvtári modulba tenni.

Nagyon fontos, hogy az esetleg rendelkezésünkre álló többféle SQL parancsváltozat közül azokat használjuk, amelyeket a standard SQL is definiál. Ilyen eltérő dialektusai vannak az INSERT, UPDATE és SELECT parancsnak is.

Lényegesen nehezebb a helyzet azonban ott, ahol a DDB/4 SQL bövitései eredetileg azért kerültek az alkalmazásba, mert azok lényegesen megkönnyitették a programozást. Ide tartozik a SELECT és FETCH parancsok { FIRST|LAST|PREV|POS() } kulcsszóval történő használata, valamint az eredményhalmazok kezelése.
Ezekben az esetekben a standard SQL utasitásokra való áttéréssel a program eredeti szerkezetét is jelentösen módositani kellett.

2. Az adattipusok és a hostváltozók eltérő kezeléséből adódó különbségek.

Az egyes adatbáziskezelők a standard adattipusokon kivül különböző DBMS-specifikus tipusokat is bevezettek. Ezek hasznosságát a felhasználói kézikönyvek ugyan hosszasan taglalják, hordozható adatbázis létrehozásakor azonban csakis a standard SQL adattipusokat szabad használni. Sajnos még ezen tipusoknál is adódnak DBMS-enként apróbb eltérések, például a karakteres mezők hosszúsága, a numerikus mezők ábrázolási pontossága vagy egyszerüen csak más a hasonló tipusok elnevezése. Ezen problémákat elsősorban rendszeradminisztrátori szinten kellett megoldani.

Az alkalmazásban viszont az ezzel kapcsolatos fő gondot az egyes adatbázistipusokhoz tartozó hostváltozók eltérő kezelése okozta. Igy a standard tipusok körét azon tipusokra korlátoztuk, amelyekhez tartozó hosváltozót az egyes DBMS-ek precompilerei hasonló módon kezelik:
CHAR, INT, LONG, FLOAT.

Nem használtuk a MONEY, DATE, TIME tipusokat, hiszen ezek adatbáziskezelőnként más-más formátumban adják vissza egy hostváltozóban az aktuális értéket. Helyettük inkább CHAR(n) tipusokban tároljuk karakteres formában a megfelelő értékeket.

A karaktertömb tipusú hostváltozóba való betöltéskor is adódnak különbségek aszerint, hogy záró-szóköz levágás történik-e vagy sem, és hogy a terminálo '\0' bekerül-e automatikusan a string végére. Az ORACLE nem tesz terminálo '\0'-t, az INGRES és INFORMIX nem vágja le záró szóközöket, a DDB/4 a záró szóközöket is levágja, valamint a stringet is terminálja.

Problémát okozott az is, hogy a különböző precompilerek másképp viselkednek bizonyos határesetekben.

Ahhoz, hogy a fenti különbözőségeket megszüntessük, készitettünk egy olyan kiegészitő eszközt, amely az eredeti precompiler és a C compiler közé ékelödve, biztositja a hostváltozók azonos kezelését. Igy tudtuk elérni, hogy a hostváltozók minden DBMS-ben az eredetileg megszokott DDB/4-es környezet szerint müködjenek.

Egy további eszköz elkészitését igényelte az alábbi probléma:
A DDB/4 megengedi, hogy az indikátorváltozókra struktúraként, vagy tömbként hivatkozzunk, INGRES-ben ezt csak tömbként, INFORMIX-ban pedig sem tömbként, sem struktúraként nem tehetjük meg. ORACLE-ben a hostváltozó sem lehet struktúra. Van ahol az indikátorváltozó csak short tipusú lehet, másutt egyéb numerikus tipus is elfogadott. A DDB/4-es alkalmazásban gyakran használtunk struktúrákat host-, illetve indikátorváltozóként. Hogy az eredeti programban minél kevesebb változtatást kelljen

eszközölni és az továbbra is jól olvasható maradjon, készitettünk egy olyan struktúra-kifejtő programot, melyet a precompiler aktivizálása előtt hivunk meg, és melynek feladata a fenti struktúrákat úgy kifejteni, hogy azt már bármelyik precompiler elfogadja.

3. A hibakezelésből adódó eltérések

1. Az azonos hibákhoz tartozó hibakódok DBMS-enkénti eltérése miatt csak szimbolikus konstansokra célszerű a hibakezelésben hivatkozni.

2. Gondot okozott továbbá az is, hogy bizonyos hibák csak bizonyos DBMS-ekben léphetnek fel, ezek lekezelése csak az adott környezetben szükséges.

3. Adott hiba esetén annak jelzése DBMS-specifikusan történhet. Ide tartozik az is, hogy az SQLCA (SQL Communication Area), amely a program felé különböző hibák és státusz információk közlésére használt terület, szintén implementáció függő.
Általában igaz, hogy a hibakódok ezen SQLCA struktúra egy adott elemében kerülnek visszadásra, ez az ANSI szabvány. Néhány eltérés azonban itt is előfordul: másképp történik - például egy sor beszúrása esetén - a DUPLICATE_KEY hiba jelzése DDB/4-ben mint INGRES-ben.

   Ahhoz, hogy az alkalmazói programokat ne kelljen emiatt módositani, az ilyen jellegű eltéréseket is az adattipusok eltérő kezelésénél emlitett kiegészitő eszközzel oldottuk meg.

4. A DEADLOCK lekezelése

   Itt is ügyelni kellett arra, hogy mind a hiba jelzése, mind az esetleges implicit akciók erösen adatbázisfüggőek lehetnek. DDB/4-ben egy inicializálási paraméter /a LOCK TIMEOUT/ beállitásával adható meg, hogy mennyi ideig lehet egy objektum lefoglalva úgy, hogy közben SQL parancsban nem hivatkozunk rá. Ha ez az idő lejár, akkor a program hibaüzenetet kap, és a tranzakcióra automatikus ROLLBACK történik, azaz visszatöltődik a tranzakció megkezdése előtti állapot.
   INGRES-ben miután az adatbáziskezelő felismerte, hogy két processz DEADLOCK szituációba került, az egyiknek DEADLOCK hibaüzenetet küld, és automatikusan egy ROLLBACK-et hajt végre.
   Az INFORMIX is felismeri a DEADLOCK szituációt, de jelzésére nincs külön hibakódja. Az SQL hibaként visszaadott kódja a LOCK_ON_UPDATE, LOCK_ON_READ vagy a RECORD_LOCKED hibakódok valamelyike. A másodlagos hibakódból amit ISAM hibakódnak neveznek, és ami csak INFORMIX környezetben létezik, tudhatja meg a program,

hogy valójában DEADLOCK történt. Automatikus ROLLBACK nem történik, annak végrehajtása a program feladata.

4. Eltérések a tranzakció-kezelésben

Néhány DBMS e területen is egyedi kiterjesztéseket hozott létre, igy itt is ügyelni kellet, hogy csak olyan parancsokat használjunk, ami mindegyikben megvan: "COMMIT WORK", "ROLLBACK WORK".

Ne használjuk viszont a "SAVEPOINT", illetve ORACLE-nél a "SET TRANSAKTION READ ONLY " parancsot.

A tranzakciók kezdetét az ANSI szabványnak megfelelően a adatbáziskezelők implicit módon ismerik fel, ez a legtöbb DBMS-nél az alapértelmezés, INFORMIX esetén azonban az adatbázis létrehozásánál kell a "MODE ANSI" opciót megadni, hogy ez a fontos feltétel teljesüljön.

Szintén a tranzakció kezelésnél okozott problémát, hogy az eredeti DDB/4-es alkalmazásban a lekérdezések eredményét egy ideiglenes belső táblában tároltuk, aminek létezése nem kötődött tranzakció-határhoz. A portálás során az ilyen eredményhalmazok helyett standard CURSOR-os olvasást kellett alkalmaznunk, itt viszont figyelembe kellett venni, hogy egy nyitott CURSOR a tranzakció végén automatikusan bezáródik. Emiatt az eredeti alkalmazást ilyen szempontból több helyen módositani kellett.

Általános érvényü szabály, hogy a tranzakció a lehető legrövidebb legyen. Ezt nem csak a tranzakció végéig tartó LOCK indokolja (ha rövidebb a tranzakció javul a konkurrencia), hanem a programok áttekinthetősége is, és könnyebb módosithatósága is. Követendő továbbá az az elv is, hogy egy tranzakción belül ne legyen interaktiv rész, hiszen ilyenkor a tranzakció vége a felhasználótól függ.

5. A CURSOR, illetve az eredményhalmaz kezelése

Néhány DBMS e téren is jelentős bövitéseket hozott létre, amelyek lényegesen megkönnyitik az adott környezetben dolgozó programozók munkáját. Ilyen a névvel ellátott eredményhalmazba történő SELECT, amelyből azután véletlenszerü hozzáféréssel választhatunk ki sorokat, és amelyekből további SELECT-et hajthatunk végre. Az eredeti alkalmazás is sürün kihasználta a DDB/4 ilyen jellegü szolgáltatásait. INFORMIX-nál ugyan más módszerrel, de funkcionálisan hasonló eredményt érhetünk el a "SCROLL CURSOR", "CURSOR WITH HOLD", és a "SELECT ... INTO TEMP táblanév" parancs használatával. Mivel ezen parancsokat a standard SQL nem ismeri, használatukról a program módositásának árán is le kellett mondanunk.

6. Adatbázisoperátorok és függvények.

Meg kellett szüntetni az összes olyan függvény és operátor használatát, amelyek a többi DBMS-ben nem ismertek, bármilyen hasznosak lennének is az alkalmazás szempontjából. Ezek közül elsősorban a string, az aritmetikai és a dátumot kezelő függvények elhagyása okozott nehézséget. Igy csak az alábbiak maradtak:

AVG/AVERAGE, MIN, MAX, COUNT, LENGTH

Sajnos még ezen függvények között is előfordulhat eltérés: A "SELECT COUNT(mezől) ..." parancs INFORMIX-ban az összes rekord számát adja meg, azokat is beszámitva, ahol "mezől" értéke NULL. DDB/4-ben csak azoknak a számát kapjuk vissza, amely rekordokra a "mezől" értéke nem NULL.

7. A tesztelés

A fenti szempontok alapján átirt, immár futóképesnek tartott programok tesztelése egy komplex alkalmazás esetén szintén komoly feladat. Ehhez DDB/4-ben egy nagyon hasznos fejlesztő eszköz - a runtime databasetrace - állt rendelkezésünkre. Segitségével egy adott program futása közben egy fájlban naplózásra kerül az összes programból kiadott SQL utasitás és azok eredménye.
Ez az eszköz az eredeti alkalmazás tesztelése során nélkülözhetetlenné vált számunkra, igy amikor kiderült, hogy nem minden adatbáziskezelő rendelkezik ilyen jellegü nyomkövető lehetőséggel, azonnal felmerült egy ilyen eszköz elkészitésének szükségessége.
Készitettünk egy rutingyüjteményt amely egy DDB/4-hez hasonló adatbázis-nyomkövetést készit INGRES, INFORMIX és ORACLE környezetben is. Ez a tulajdonság már forditási időben beépül az egyes programokba, és az alkalmazás elindítása előtt egy környezeti változóban lehet beállitani, hogy készüljön-e nyomkövetés vagy sem.

III. DBMS-független problémák

A fentiek voltak tehát azok az általános problémák, amelyekekkel az eltérő adatbáziskörnyezet miatt kellett megküzdeni.
Érdemes azonban megjegyezni, hogy az alkalmazás átültetésekor nemcsak a DBMS változott, hanem az operációs rendszer (különböző UNIX változatok), a C-compiler, illetve a hardver is. Ezen eltérésekből adódott tipikus hibák megemlitése is hasznos lehet minden olyan projekt számára, ahol a hordozhatóság fontos szerepet játszik:

- A UNIX könyvtári rutinok különbözősége.
  Ez elsősorban az AT&T, illetve Berkeley UNIX verziók között okozott gondot. Bizonyos rutinok csak az egyik

verzióban vannak meg, mások nem egyformán működnek. Hasonló gond volt, hogy a standard UNIX "/usr/include" katalógusban levő "*.h" fájlok esetenként eltérő tartalommal rendelkeznek.

- C compilerek különbözősége is számos problémának volt a forrása. Nem minden compiler felelt meg az ANSI C szabványnak, igy gyakori eset volt, hogy az egyik compiler által hibátlannak ítélt programra egy másik compiler számos szintaktikai hibát jelzett, - vagy ami még rosszabb - esetleg "csak" másképp működő kódot generált.

- Az ilyen jellegü programhibák oka gyakran az eltérő hardverarchitektúrából adódott:
  - azonos tipusok eltérő mérete;
  - bájtok szavakon belüli eltérő sorrendje;
  - bizonyos tipusok adott cimen kezdődhetősége;
  - eltérő előjelkiterjesztés;

Ezen tipusú hardver és szoftver függőségek megszüntetése az alkalmazásban szintén nélkülözhetetlen feltétele volt a hordozhatóság megteremtésének. Ennek fő eszköze elsősorban az ANSI C szabvány betartása valamint a megfelelő C-compilerek ANSI opcióval történő használata volt. Sajnos néhány eseteben egyedi megoldás, a forrás módositása, vagy "#ifdef ..." makrók használata vált szükségessé.

IV. Összefoglalás

Az eddig elmondottakat összegezve az adott feladat kapcsán az alábbi konkluziók tehetők:

- Egy C-be ágyazott SQL interfészt használó alkalmazás egy másik DBMS környezetbe történő átültetése bár nem triviális, de végrehajtható vállalkozás.

- Az adott DBMS-ek adminisztrátori szintü ismeretére feltétlenül szükség van.

- Ismerni kell a DBMS-hez tartozó precompilerek sajátosságait.

- Az ANSI SQL ismerete is feltélenül szükséges, sajnos azonban nem elégséges feltétel. Használata mindazonáltal erősen ajánlott.

- Az alkalmazásban kikerülhetetlenül megmaradt DBMS-specifikus SQL hivások közös forrásba tételével célszerü lokalizálni azon parancsok, illetve kódrészek körét, amelyeket újabb portálás esetén esetleg módositani kell.

Osztott adatbázis megvalósításának lehetőségei az INFORMIX RDBMS keretében.

Paál Péter-KFKI TRADIS Kft.

1. Bevezetés.

A korszerű adatbázis alkalmazások megkövetelik, hogy az alkalmazások a meglévő hardver konfiguráción szétoszthatók legyenek. Ez nem csak azt jelenti, hogy az egyes alkalmazások a különböző gépekről elérjék az adatbázist, hanem azt is, hogy a tárolt adatok fizikailag különböző gépeken helyezkedjenek el. Erre akkor van szükség, ha már meglévő adatbázisokat szeretnénk összekapcsolni, vagy az adatbázis növekedésekor a rendszert további számítógéppel szeretnénk bővíteni. Ez utóbbi esetben az alkalmazás korlátja nem marad a gép teljesítménye, hiszen egy újabb adatbázis szervert kapcsoltunk a rendszerbe. A jövőben az osztott adatbázisok megadják a lehetőséget, hogy a fontos adatokat két független gépen duplikáljuk, így az egyik hibája esetén a másikról elérhető a teljes adatbázis.

Osztott adatbázisok esetében az adatbáziskezelő rendszernek kell megoldani, hogy a felhasználó az alkalmazás szempontjából ne tudjon különbséget tenni, hogy az adatbázis a helyi gépen van-e, vagy pedig valamilyen hálózati eszköz segítségével távoli gépről érhető el. Szintén a rendszer által kell támogatni, hogy az egyes tranzakciók folyamán a teljes osztott adatbázis mindenkor konzisztens legyen, mindenkor megőrizze integritását.

2. Autonóm adatbázis szerverek.

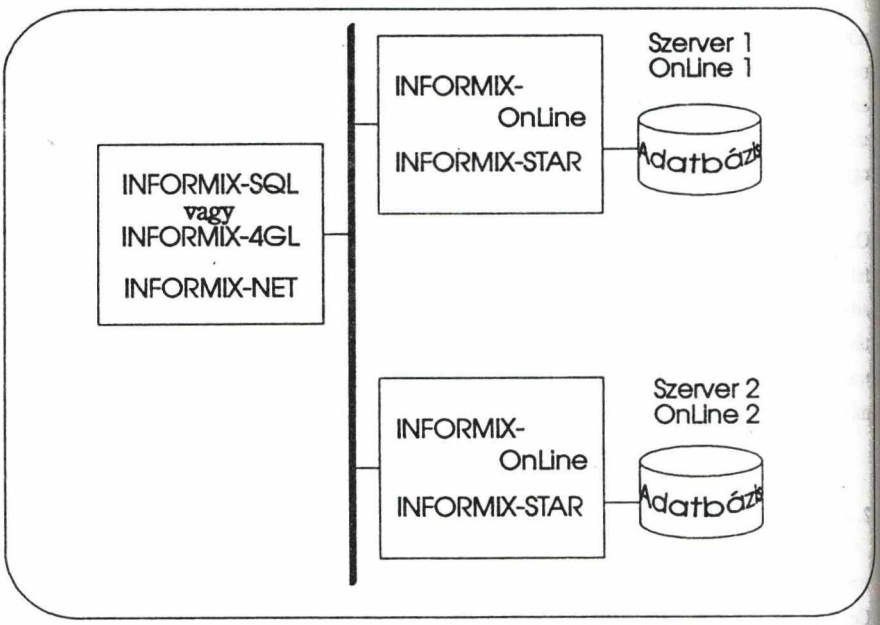
Az osztott adatbáziskezelő rendszerek lehetnek centralizáltak vagy teljesen osztottak. Centralizált esetben az adatbázis szerverek között van egy kitüntetett, amely a szerverek együttműködését koordinálja. Ebben az esetben a központi szerver hibája megbénítja a teljes rendszeren az adatbázisok kapcsolatát. Az alkalmazás szempontjából a központi szerver lehet a korlát, hiszen az összes SQL utasítás először a központi szerverhez kerül, és csak azután jut el a megfelelő adatbázis szerverhez.

Az INFORMIX relációs adatbáziskezelő rendszer az osztott adatbáziskezelést teljesen osztott formában valósítja meg, vagyis minden adatbázis szerver egymástól

teljesen független (site autonomy). A független adatbázis szerverek egymástól függetlenül kezelik a saját helyi adatbázisukat, irányítják a saját lockolási, loggolási és recovery mechanizmusokat, és így nem függnek semmilyen központi szervertől. Természetesen egymással együttműködnek ha kell (ld. Two phase commit).

2.1 Autonóm adatbázis szerverek INFORMIX termékekkel.

Az INFORMIX termékekkel megvalósított osztott adatbázis rendszer látható az 1. ábrán.



1. ábra
INFORMIX osztott adatbázis

A rendszernek két független adatbázis szervere van, amelyek a SZERVER1 és a SZERVER2 gépen működnek. Mindkét szerveren az INFORMIX-OnLine az adatbázis meghajtó (backend), amelyeknek példánkban az online1 és online2 nevet

adjuk. Az osztott adatbáziskezelést az OnLine-hoz kapcsolódó INFORMIX-Star valósítja meg.
Az alkalmazás kliens példánkban INFORMIX-SQL (SQL nyelv értelmező szoftver kiegészítve form és riport generáló funkciókkal) és/vagy INFORMIX-4GL (4GL nyelv fordító és futtató). Az alkalmazás az INFORMIX-Net szoftver segítégével kapcsolódik az adatbázis szerverekhez.

Meg kell jegyezni, hogy az alkalmazás futhat valamelyik szerver gépen is, és abban az esetben az alkalmazáshoz nem kell INFORMIX-Net.

A következő hálózati protokollokat támogatja az INFORMIX rendszer:

-TCP/IP Transport Layer Interface (TLI) vagy
standard Berkley socket hívások
-StarGROUP 3.1, 3.2, 3.3 vagy 3.4 TLI hívásokkal
-Felhasználói protokoll, speciális hálózati
driver szükséges

3. INFORMIX adatbázis szerverek konfigurálása.

Mielőtt az INFORMIX rendszer konfigurálása megkezdődhet, a szerver gépet kell megfelelően konfigurálni a hálózathoz. Ez a következő file-okat érinti TCP/IP hálózat esetében:

- /etc/hosts
- /etc/hosts.equiv vagy .rhosts
- /etc/services

A /etc/hosts file tartalmazza a hálózaton elérhető hostok internet címét, a host megnevezesét, és az esetleges alias megnevezést. A /etc/hosts.equiv file tartalmazza a "bizalmas" szervereket, amelyeken azonos nevű felhasználók dolgoznak. Az INFORMIX rendszer esetében az összes adatbázis szerver gépének szerepelnie kell ebben a file-ban.

Példa a /etc/hosts és /etc/hosts.equiv és /etc/services file-ra:

/etc/hosts

192.9.1.20 SZERVER1
192.9.1.30 SZERVER2

/etc/host.equiv

SZERVER1
SZERVER2

/etc/services

star1 1541/tcp
star2 1542/tcp

A /etc/services file adja meg azt az információt, amely a hálózaton elérhető szolgáltatásokat a megfelelő port számmal és protokollal együtt ismertté teszi a hálózat gépei számára. Igy a kliens és szerver processzek tudják, hogy mely porton, milyen protokollal kell kommunikálniuk egymással a hálózaton. Ennek a file-nak minden adatbázis szerveren jelen kell lennie, és ugyanazt az információt kell tartalmaznia. Minden adatbázis szerverhez (backend) kell rendelni egy sort ebben a file-ban.

A $INFORMIXDIR/etc/sqlhosts file a TCP/IP illetve a StarGROUP hálózati szoftverek számára nem elérhető, az az INFORMIX rendszer számára fontos. (A $INFORMIXDIR az a könyvtár, amelyik az INFORMIX rendszerszoftvert tartalmazza.) Ezt a file-t használja a rendszer, hogy azonosítsa az OnLine rendszert (adatbázis szerver) a hosttal, amelyiken az OnLine szoftver fut, továbbá a hálózati szolgáltatást a hoston.

Példa az sqlhosts file-ra:

szerver név	**hálózat típus**	**host név**	**szolgáltatás**
online1	tcp	SZERVER1	star1
online2	tcp	SZERVER2	star2

A SZERVER1 gépen online1 nevű OnLine rendszer fut INFORMIX-Star-ral, a SZERVER2 gépen online2 OnLine rendszer fut INFORMIX-Star-ral. Igy mindkét gép adatbázis szerver, az alkalmazások mindegyik gép adatbázis szerverét elérik.

A következő lépés a konfigurációban, hogy a szerver gépeken elindítjuk az sqlexecd démonokat, amelyek figyelik, hogy az adott porton érkezik-e kérés a szolgáltatás eléréséhez. Az sqlexecd démont a /etc/services file megfelelő szolgáltatásával kell elindítani.

pl.
$INFORMIXDIR/lib/sqlexecd star1
a SZERVER1 hoszton.

Az sqlexecd megnézi a /etc/services filet, és megnézi, hogy a star1 szolgáltatáshoz melyik porton milyen protokolt kell figyelni.

Nézzük meg ezek után, hogy hogyan jön létre a kapcsolat a kliens gép és a szerver gép között.

A kliens gép adatbázismeghajtója (pl. online2 a SZERVER2 gépen) az alkalmazástól kap egy SQL utasítást, amelyik a SZERVER1 gép adatbázis szerverére (online1) vonatkozik.

DATABASE d1@online1 (a d1 adatbázis az online1 rendszerben)

Az online2 meghajtó megnézi a $INFORMIXDIR/etc/sqlhosts file-ban, hogy az online1 adatbázis szerver melyik szerveren van (SZERVER1), milyen hálózattal érhető el (tcp), és milyen szolgtatás tartozik hozzá (star1). Ezek után megnézi a /etc/hosts file-t, hogy a talált szerver névhez milyen Internet cím tartozik (192.9.2.20). Hogy kommunikálni tudjon a hálózaton, szükség van a hálózati port címre, és a protokollra. Ezt tudja meg a /etc/services file-ból. Ilymódon létrejött a kapcsolat a kliens és a szerver gép között, hiszen a szerver oldalon az sqlexecd démon figyeli a kiválasztott portot és protokollt.

4. Távoli adatbázisok használata.

4.1 SQL kiterjesztése.

Ahhoz, hogy több adatbázist érjünk el egyszerre, az SQL terminológia kiterjesztésére is szükség van.

a. Egy keresés (query) vonatkozhat olyan táblákra is, amelyek nincsenek az aktuális adatbázisban. Ez a lehetőség akkor is adott, ha a felhasználóknak nincs lehetőségük hálózati kommunikációra (pl. azonos adatbázis szerveren). Azt a táblát, amelyik nem az aktuális adatbázisban van, külső táblának nevezzük.

b. Az aktuális adatbázis lehet távoli adatbázis szerveren is.

c. A külső táblák lehetnek olyan adatbázisokban, amelyek nem az aktuális adatbázist tartalmazó szerveren vannak. Ez azt jelenti, hogy kiadható olyan SELECT utasítás, amelyik olyan táblákra vonatkozik, amelyek külböző adatbázisokban, különböző szervereken vannak.

4.2 Az aktuális és a távoli adatbázisok típusai.

Az Osztott adatbázis környezetben az adatbázisoknak ugyanolyan típusúaknak kell lenniük. Ez azt jelenti, hogy olyan adatbázis, amely nem használ tranzakció naplózást (no logging, az INFORMIX rendszer ilyen típust is megenged) csak olyan adatbázist érhet el, amelyik szintén nem használ tranzakció naplózást. Ha az aktuális adatbázis használ naplózást, csak olyan adatbázissal működhet együtt, amelyik szintén használ tranzakció naplózást.

MODE ANSI és non-MODE ANSI adatbázisok szintén nem tudnak együttműködni.

Az INFORMIX rendszer az adatok konkurens hozzáférésére négyféle szintet enged meg (ISOLATION LEVEL; dirty read, committed read, cursor stability, repeatable stability). Az aktuális adatbázisnak és a távoli adatbázisnak az elkülönülési szintjének ugyanolyannak kell lennie.

4.3 Külső táblák megnevezése.

Osztott adatbázis környezetben a táblákat a következőképpen lehet azonosítani:

adatbázis_név@szerver_név:[tulajdonos].tábla_név,

ahol az **adatbázis_név** az adatbázis neve a távoli szerveren, **szerver_név** az adatbázis szerver neve (pl. online1), **tulajdonos** a tábla tulajdonosa, (opcionális, nem mindig kötelező), **tábla_név** a tábla neve.

Példaként tekintsünk egy SELECT kifejezést, amelyik két különbő adatbázis táblájára vonatkozik:

```
SELECT order_num, lname, fname FROM
        masterdb@online1:customer C,
        sales@online2:orders O
    WHERE
        C.cust_no = O.cust_no
```

Az aktuális adatbázis meghajtó kommunikál az **online1** és **online2** adatbázis szerverekkel. A SELECT kifejezésben használtuk a C és O helyettesítéseket, hogy rövidítsük a WHERE kifejezést.

4.4 Táblák helyettesítése. A Synonym fogalma.

A külső táblák előbb látott hosszú elnevezése helyett célszerű helyettesítő név használata, amelyet synonym-nek nevezünk. A synonym egy táblára, vagy view-ra vonatkozhat, és a CREATE SYNONYM utasítással hozzuk létre.

CREATE SYNONYM rövid_név FOR hosszú_név

Abban az adatbázisban, amelyben a synonym-et definiálták, a synonym táblaként látszik, és teljesen úgy viselkedik, mint egy közönséges tábla. Ez azt jelenti, hogy a

felhasználó a synonym táblát úgy használja, mintha lokális tábla lenne, azaz a tábla fizikai elhelyezkedése az alkalmazás szempontjából teljesen transzparens.

Fontos szabály, hogy synonym nem alkotható egy másik synonym-ra, tehát a CREATE SYNONYM végrehajtásakor a táblának, vagy view-nak valóságosnak kell lennie az adatbázisban. Időközben azonban lehetőség van arra, hogy a valóságos táblát áthelyezzük egy másik adatbázisba, (például ha nincs elég tárolási kapacitásunk a gépen), viszont a tábla eredeti helyén elhelyezünk egy synonym-át a tábla eredeti nevével, amelyik most az áthelyezett táblára vonatkozik. Mivel a synonym futás közben konvertálódik valódi tábla névvé, ha kiderül, hogy a tábla helyén már egy másik synonym áll, a konvertálódás folytatódik. Ez a synonym lánc 16 részből állhat, és mivel synonym-ot nem lehet synonym-ra definiálni, a lánc nem záródhat be.
Igy a felhasználó észre sem veszi, ha a használt tábla időközben átköltözik egyik gépről a másikra.

5. Integritás megőrzése több adatbázist egyszerre érintő tranzakciók esetében.

5.1 UPDATE egyszerre több adatbázis szerveren.

Az INFORMIX rendszerrel lehetőség van egy tranzakción belül több adatbázis szerver bevonására. Ez az SQL BEGIN WORK és COMMIT WORK parancsok közé zárt SQL utasításokra vonatkozik, amelyek mindegyike vagy végrehajtódik, vagy visszagörgetésre kerül (ROLL BACK).

Rendszerhiba esetén, amennyiben több szerver is bekapcsolódott a tranzakcióba, speciális helyreállítási eljárás gondoskodik arról, hogy az összes adatbázis egymással konzisztens maradjon.

5.2 Two-Phase Commit (TPC) protokol.

A TCP protokol a megfelelő eszköz arra, hogy a több szervert érintő tranzakció után az összes adatbázis konzisztens maradjon.

A TPC protokol esetében a résztvevő adatbázis szerverek között lesz egy koordinátor, mégpedig az a rendszer, amelyik az aktuális adatbázist tartalmaza. Ez a koordinátor irányítja a TPC protokollt.

A TPC protokol két fázisból áll, és mindegyik fázisnak több lépése van. Az első fázis, az un. Pre-Commit fázis a COMMIT WORK utasítással kezdődik. Első lépésben a koordinátor felkészül a lokális végrehajtásra, és egy BEGREP tételt ír a naplóba (log). Ebben a tételben szerepel az összes, a tranzakcióban résztvevő szerver is. Második lépésben a koordinátor küld egy előkészítő üzenetet a szervereknek, hogy lehetséges-e COMMIT. A résztvevők, ha a COMMIT lehetséges, egy PREPARE üzenetet írnak a saját naplóba. Ha ez megtörtént, akkor a résztvevők visszaküldik a koordinátornak, hogy készek a COMMIT-ra.

A következő fázis a Post-DECISION fázis. Első lépése a tulajdonképpeni döntés. Ha minden résztvevő visszaküldte, hogy képes a COMMIT-ra, akkor a koordinátor ír egy COMMIT tételt a naplóba, amivel jelzi a helyi commitot, és hogy a protokol elérkezett a Post-DECISION fázisba. Második lépésként a koordinátor minden résztvevőnek elküldi, hogy hajtsa végre a tranzakciót. A résztvevők mindegyike egy COMMIT tételt ír a naplóba. A harmadik lépésben a koordinátor megvárja, amíg mindegyik résztvevőtől megérkezik az üzenet, hogy committálta a tranzakciót, Ha megérkezett az összes üzenet, a koordinátor a naplóba ír egy ENDTRANS tételt. Ha ez már a naplóban szerepel, a tranzakciónak vége.

5.3 Two-Phase Commit Recovery.

Ha bármelyik adatbázis szerveren hiba történik mielőtt a TPC protokol befejeződne, az osztott adatbáziskezelő rendszernek vissza kell állítani a konzisztens állapotot. Az INFORMIX rendszer esetében a visszaállítást nem a felhasználói programnak kell elvégeznie, hanem a rendszer automatikusan visszaállítja az uncommitted tranzakciókat, illetve végrehajtja a végrehajtott (committed) tranzakciókat.

IRODALOM:

1. Dusan Petkovic: INFORMIX; Das relationale Datenbanksystem mit INFORMIX OnLine. ADDISION-WESLEY Publishing Company 1991.
2. DATENBANK EXTRA: INFORMIX: Mit offenen Systemen zu unternehmensweiten Lösungen. IT Verlag 1992.
3. INFORMIX technikai rendszerleírások
   Module 1573/92,
   Module 1061/92,
   Module 1150/92.

INGRES fejlesztési tapasztalatok

Előadó: Horváth Tibor (FreeSoft)

Tartalom:
1. Bevezető
2. INGRES/Knowledge Management
3. INGRES/Vision

1. Bevezető

Az INGRES Corporation egyike a világ első számú szállítóinak az RDMBS területen.

Az INGRES alapítása óta kizárólag a relációs adatbáziskezelő rendszerekkel foglalkozik, és a felhalmozott ismereteket kiemelkedően hasznosítja a UNIX operációs rendszerű gépek világában. Különösen fontos ez manapság, amikor minden hardware gyártó megjelenik a többproceszoros gépeivel amihez az INGRES adatbázis-szerver arhitektúra legjobban illeszthető, mivel minden CPU-hoz egy RDBMS szervert kapcsolhat, így sokkal gyorsabbak a válaszidők, mivel minden RDBMS szerver egy teljes CPU-t tud használni.

Az INGRES olyan adatbáziskezelési lehetőségeket biztosít, amelyeket ilyen komplexen más RDBMS termékeknél nem lehet megtalálni: képernyőn megjelenő adminisztrációs segédprogramok ■ a teljesítmény hangolása és figyelése ■ dinamikus diszk- terület- foglalás ■ opcionális adattömörítés ■ nagy mennyiségű adatok gyors betöltése standard adatfile-okból ■ adatok hozzáférés elleni védelme ■ rugalmas naplózás: az adatbázis fizikai és logikai sérülések elleni védelme ■ az egyes táblák vagy a teljes adatbázis mentése akár on-line üzemmódban is ■ táblák egyedi hozzárendelése diszkekhez az I/O teljesítmény javítására ■ új tranzakciókezelési technológiák, melyek a hagyományos RDBMS-rendszerekhez képest még inkább növelik a rendszer adatbiztonságát (fast commit, két fázisú commit), ...

INGRES alapelemek:
- INGRES JoinDefs
- INGRES/Forms, "VIFRED" (Visual Forms Editor)
- INGRES/QBF (Query By Forms)
- INGRES/RBF (Report by Forms)
- INGRES/REPORT WRITER
- INGRES/VIGRAPH
- Interaktív SQL

INGRES opciók:
- INGRES/Windows 4GL
- INGRES/Object Management

Jelen előadásban az INGRES két további elemét a Knowledge Management-et és a Vision-t mutatjuk be.

2. INGRES/Knowledge Management

Ismertetőnk az INGRES Knowledge Management azon jellemzőit mutatja be, amelyek biztosítják a referenciális integritást, végrehajtják a tárolt üzleti szabályokat és adatbázis riasztásokat, ill. ellenőrzik az erőforrás- és hozzáférési engedélyeket.

Az adatkapcsolatok ismerete, az alapvető üzleti szabályok, az események figyelése, az ellenőrzött felhasználói hozzáférések és az elfogadható erőforrás-szükséglet becslése korábban kívül estek a relációs adatbáziskezelők "látókörén" és - ha nem hiányoztak teljesen - kezelésük általában felületes volt.

Az adatkezelésnek ezek a hiányosságai csökkentették az alkalmazásfejlesztők hatékonyságát: a programozóknak minden egyes felhasználói programba be kellett programozniuk az adatbázis integritást és az üzleti szabályokat. Ezek a problémák a szerver-kliens hálózatok elterjedésével válnak egyre égetőbbé. Régebben, ha egy üzleti szabály megváltozott, a felhasználói programot egyszerűen lecserélték egy új verzióra. Manapság ez az eljárás vagy nagy fáradságot igényel vagy pedig csaknem lehetetlen, hiszen a szerver-kliens hálózatok növekedése miatt minden egyes kliens állomáson (10-től...1000-ig) kellene a programot cserélni.

Az INGRES megoldja ezeket a problémákat az adatbázis-szerveren belül. A következő rendszerek használatával kezeli a változásokat.

szabálykezelő rendszer
adatbázis riasztások
erőforrás-kezelés
hozzáférési jogosultság kezelése

Szabálykezelő rendszer

Az INGRES szabályok olyan procedúrák, amelyek automatikusan biztosítják az üzleti szabályok végrehajtását és a referenciális integritást. Az adatbázis-rendszer akkor hívja meg ezeket a procedúrákat, amikor egy felhasználó által definiált kritérium teljesül. A szabályokat egy új SQL paranccsal a CREATE RULE-lal lehet létrehozni. A hozzárendelt eljárás egy standard adatbázis-procedúra, amelyet az INGRES/4GL és az SQL segítségével lehet megírni. A moduláris felépítés miatt több szabály hívhatja ugyanazt az adatbázis-procedúrát. Az INGRES szabálykezelő rendszere a kritériumok két különböző típusát tartalmazza:

Referenciális integritás

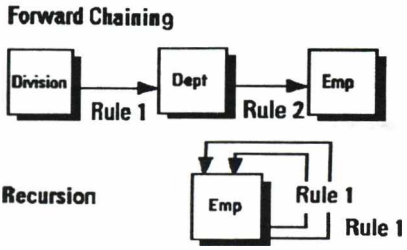
Biztosítja az elsődleges/másodlagos kulcsok egyezését. (Pl.: egy vállalati információs rendszerben egy új dolgozót csak egy már létező osztályba lehet felvenni.)

Üzleti szabályok

Alapvető megkötések, korlátozások, amelyekkel egy adott gazdasági szervezet működik. (Pl.: egy ipari alkalmazásban a 23X5A alkatrészt újra kell rendelni, amikor a készlet egy bizonyos szint alá csökken.)

A szabálykezelő korlátlan előreláncolást és rekurzív hívást enged meg, ellentétben más adatbázis-kezelő megoldásokkal, ahol ezek erősen korlátozottak. A központi szabálykezelés lehetőséget ad az

adatbázis-adminisztrátornak (DBA), hogy a megváltozott követelményeknek megfelelően gyorsan módosítsa a szabályokat. További előnye, hogy a felhasználói programok készítőjének nem kell megtanulnia és minden egyes programban lekódolnia a megváltozott szabályokat.



A programozott szabályok, amelyek közvetlenül az adatbázis-szerverben futnak, felszabadítják a programozót, csak a feladatára kell koncentrálnia, nem kell törődnie az üzleti szabályokkal és az integritás ellenőrzésével.

Az INGRES szabálykezelő jelenleg az egyetlen, amely az alábbiakat biztosítja:

- Korlátlan számú olyan független szabály definiálását egy adattáblához, amelyek moduláris felépítésükkel biztosítják az üzleti szabályok megvalósítását.
- Az értéktől függő szabály-aktiválást. A szabály csak akkor aktiválódik, ha egy vagy több mező értéke bizonyos kritériumoknak megfelelően változik.
- Korlátlan előre láncolást. Biztosítja az adatbázis integritását azzal, hogy a végrehajtott szabályok által kiváltott módosítások újabb szabályokat aktiválhatnak.
- Korlátlan számú rekurzív hívást. Ez is az adatbázis integritást biztosítja, mert a végrehajtás alatt álló szabály képes arra, hogy újra hívja önmagát.
- Az SQL/4GL megvalósítás növeli a programozó hatékonyságát, mert a procedúrák, amelyek megvalósítják a szabályt nem egy speciális nyelven, hanem az INGRES saját nyelvén, 4GL-ben írhatók.

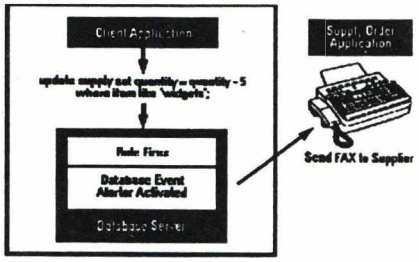
Adatbázis riasztások

Az INGRES az első relációs adatbáziskezelő, amely SQL utasítások használatával párbeszédet biztosít az adatbázis és a felhasználói programok között. Az adatbázis riasztás funkció egy egyszerű eszköz arra, hogy az eseményekről - amelyek a mindennapi élet hozzátartozói - a felhasználó értesítést kapjon, és megfelelő módon válaszoljon ezekre.

Az adatbázis riasztások olyan SQL utasítások, amelyek definiálják és érvényesítik azokat az eseményeket, amelyek az adatbázisban vagy a felhasználói programban bekövetkeznek. A felhasználói program vagy az adatbázis üzenetet tud küldeni egy másik felhasználói programnak, jelezve azt, hogy egy

meghatározott esemény bekövetkezett. Az esemény feldolgozásakor bármilyen típusú választ lehet kezdeményezni, amely magában foglalhat adatbázis- vagy nem adatbázis műveleteket is. Ez a megoldás engedélyezi, hogy a szerver üzenjen egy kliensnek. Ezzel - akár helyi akár távoli programról van szó - sokkal aktívabb szerepet jut a szervernek a programok kezelésekor.

Az adatbázis-események a következő három SQL paranccsal hozhatók létre: CREATE DBEVENT, REGISTER DBEVENT, RAISE DBEVENT. A programok, melyek az eseményeket aktiválják INGRES/4GL-ben vagy 3GL-ben írhatók meg. Mivel az adatbázis riasztások az intelligens adatbázis részét képezik, az INGRES szabályok aktiválhatnak eseményeket. Pl.: egy adatbázis-módosítás beindít egy szabályt, mert a mennyiség az újrarendelési szint alá esett. A szabály aktiválja az adatbázis riasztást, amely üzen egy programnak, hogy küldjön FAX-on megrendelést a beszállítóhoz. Mindez teljesen automatikus, nincs szükség felhasználói beavatkozásra.



Az INGRES adatbázis riasztásokat számos helyen lehet alkalmazni. Néhány példa:

- Automatikusan generálhatunk egy listát a késedelmesen fizetőkről, amikor a fizetési kötelezettség határideje pl. 5 napja lejárt.
- Ha egy felhasználó illegális hozzáférést kísérel meg az adatbázishoz, a rendszer azonnal üzenetet küldhet az adminisztrátor képernyőjére.
- Amikor a napi költségvetési adatok beérkeztek a pénzügyi rendszerbe, a management számára automatikusan lefuttatható egy riport-generálás az új költségvetésről.

Az adatbázis riasztások helyettesítik azokat a meglévő eljárásokat, amelyeket eddig csak bonyolult programok segítségével lehetett megvalósítani. Ezek állandóan foglalták a rendszer erőforrásait, hogy figyeljék ha valami változás történt az adatbázisban. Az INGRES adatbázis riasztásokat használva csak akkor aktiválódik egy szabály, ha egy esemény bekövetkezett.

Erőforrás-kezelés

Az INGRES erőforrás kezelője együttműködik az INGRES kérés optimalizálójával. Korlátozza a szerver erőforrás-használatát, a felhasználói kérés erőforrás-szükségletének figyelembevételével. Így biztosítja, hogy a rendszer teljesítménye ne csökkenjen, ha a felhasználó kiad egy "elfutó" (runaway) kérést. A statiszti-

kai alapú optimalizáló a kérés végrehajtása előtt megbecsüli a kérés teljesítéséhez szükséges visszaadandó sorok és a diszk I/O-műveletek számát. Egy nagy tábla rendezéséhez több százezer diszk I/O-ra lehet szükség még mielőtt az első sort vissza tudná adni. Mikor az INGRES DBA beállít egy határértéket a visszaadott sorok és diszk I/O-műveletek max. számára, az erőforráskezelő rendszer ellenőrizni fogja a határértékeket, és még a feldolgozás kezdete előtt abortálja a kérést.

Hozzáférési jogosultság kezelése

Az adatokhoz való speciális hozzáférést engedélyező rendszer magában foglalja a csoport-jogosultságot és a felhasználói programokhoz való hozáférés kezelését, bizonyos adatvezérlési utasítások engedélyezését, és lehetőséget biztosít az erőforrások korlátozására a csoportok számára. A többszörös, különböző szinten megvalósított adathozzáférési engedélyek, az adatbiztonság megsértése nélkül, csak a szükséges adatokhoz való hozzáférést engedélyezik.

A DBA csoportengedélyeket adhat egy felhasználói csoportnak, hogy az pl. csak a SELECT utasítást hajthatja végre. Az INGRES rendszerben a csoport a felhasználók halmazát jelenti, melybe könnyen felvehető vagy belőle törölhető egy felhasználó.

A "szerepkör-engedélyek" az adatokhoz való hozzáférést egy speciális alkalmazásra vonatkozóan szabályozzák.

Az adatvezérlési nyelv utasításaihoz mint pl.: CREATE TABLE, SET LOCKMODE is hozzárendelhetők felhasználói jogosultságok. Az adatbázis-rendszer ellenőrzi a jogosultságot, mielőtt végrehajtaná az utasítást.

Az INGRES adatbázis-szerver a Knowledge Management kiterjesztéssel jelentősen növeli a programozói hatékonyságot, és sokkal nagyobb ellenőrzést biztosít a szükséges adatbiztonság megőrzése érdekében.

3. INGRES/Vision

Költségvetés-csökkentés
kisebb programozói kapacitás
nagy felhasználói programok
jelentős erőforrások hozzárendelése egy már létező rendszer
karbantartásához

jelentik azt, amit egy mai korszerű gazdasági szervezetnek figyelembe kell vennie, ha növelni akarja programozóinak hatékonyságát.

Az INGRES/Vision az a programkód-generátor, amely biztosítja, hogy a már meglévő hardware eszközökön olyan stratégiai üzleti rendszer valósuljon meg, amelynek kifejlesztése sokkal kevesebb időt vesz

igénybe, mintha más felhasználói programfejlesztő eszközöket használtunk volna.

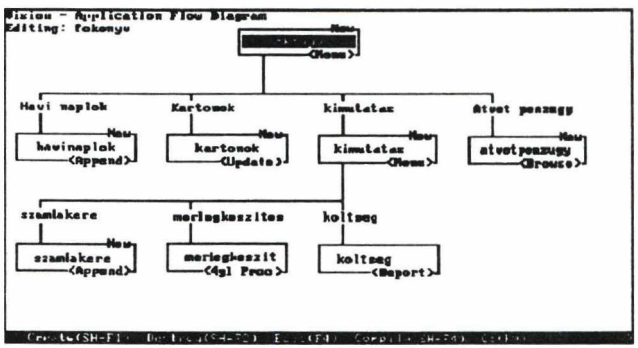
Az INGRES/Vision előnyei

Az INGRES/Vision nagy előrelépést jelent a kódgenerálási technológiában sokkal inkább növeli a termelékenységet mint egy egyszerű fejlesztő program. A Vision-nel kifejlesztett programok a felhasználói rendszer bonyolultsága ellenére is könnyen karbantarthatók. Ezt a Vision felhasználói programkezelő architektúrája teszi lehetővé.

A Vision egyedi architektúrája a felhasználói program szerkezetének vizuális megjelenítésével együtt biztosítja Önnek a fejlesztési és a program-karbantartási idő jelentős csökkenését.

A Vision standard része a továbbfejlesztett - különösen a fejlesztő termelékenységét biztosító - felhasználói interface, amely magába foglalja:

- az alkalmazás szerkezetének képi megjelenítését
- menükezelési lehetőségeket
- on line help rendszert
- pop-up listák generálását már létező értékekre



Alkalmazás-fejlesztés INGRES/Vision-nel

1. lépés

Az INGRES/Vision környezetben két fő rész van: az Alkalmazás szerkezeti diagram és a Vizuális kéréseditor. A két funkció használatával az alkalmazás-fejlesztés és - karbantartás egyszerű folyamat.

Az Alkalmazás szerkezeti diagramot úgy használhatjuk a program struktúrájának a leírására, mintha egy szerkezeti vázlatot készítenénk. Először specifikáljuk a képernyő-terveket (kereteket) és a menüpontokat,

amelyeken keresztül a felhasználó elérheti a szükséges kereteket. A keretek létrehozásakor mindig definiálni kell a keret típusát (Menü, Módosítás, Felvétel, Lekérdezés), és azt, hogy milyen adattáblákat fog kezelni.

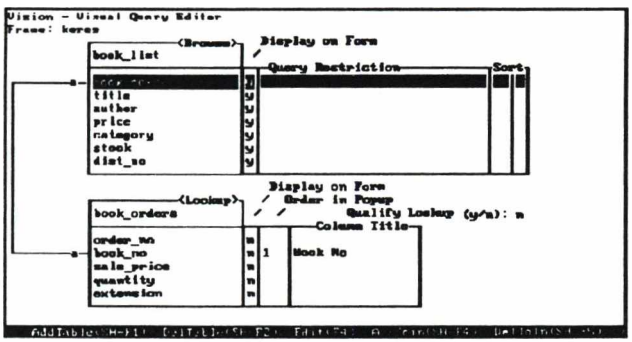
A Vizuális kéréseditor használatával definiáljuk, hogy milyen adatokhoz akarunk hozzáférni, és milyen műveleteket engedélyezünk a felhasználónak. A keretek és az adatspecifikációk megadásával az INGRES/Vision automatikusan generálja az alkalmazás képernyő-terveit, beleértve a formátumokat, menüket, lekérdezéseket és a hibamentes 4GL programkódot.

2. lépés / Az alkalmazás továbbfejlesztése

Az INGRES/Vision automatikusan generálja az összes szükséges kódot, amely egy átlagos alkalmazás felépítéséhez szükséges. Azonban az INGRES/Vision - eltérően a legtöbb kódgenerátortól - támogatja, hogy Ön anélkül hajtson végre jelentős továbbfejlesztéseket a programján, hogy kézzel módosítani kellene a legenerált kódot. Ez azt jelenti, hogy Ön kihasználhatja a Vision kódgeneráló lehetőségeit és jellemzőit úgy, hogy az alkalmazását az INGRES 4GL nyelv használatával fejleszti tovább.

Például Ön megteheti, hogy

- Kiír egy üzenetet, amikor a felhasználó meghív egy keretet, belép egy mezőbe, vagy kiválaszt egy menüfunkciót;
- Nyilvántarthatja hogy hány rekord került lekérdezésre, módosításra, felvételre;
- Beállíthat "if-then" feltételeket a hívó keretek számára vagy más tevékenység végrehajtására;
- Hozzárendelhet kezdeti értékeket egy globális vagy egy lokális változóhoz.



3. lépés / Összetett, bonyolult alkalmazások készítése

Hogy Ön könnyen kezelje ad-hoc jellegű kéréseit a Vision architektúra még a legbonyolultabb alkalmazások készítésénél is, biztosítja a szükséges rugalmasságot és teljesítményt.

A Vision támgatja:

- 4GL-, 3GL- és adatbázis-procedúrák kezelését
- automatikusan generált szekvenciális kulcsokat
- adatok cseréjét a keretek között
- kiszámított mezőket
- globális és lokális változókat
- team-munkát

Az INGRES/Vision környezetének megváltoztatása

Hogy az Ön programjának használhatóságát növeljük, a Vision lehetőséget nyújt arra, hogy Ön - különleges igényeinek megfelelően - módosítsa a legenerálandó kódot. Például:

- Megváltoztathatja a Vision által generált menük sorrendjét és szövegét.
- Automatikusan hozzátehet egy tetszőleges menüpontot.
- Megváltoztathatja az INGRES/Vision által kiírt üzenetek szövegét.
- Módosíthatja a hibakezelést, amelyet a Vision hajt végre egy lekérdezés után.

Kapcsolat más INGRES-eszközökkel

Az INGRES/Vision alkalmazások szoros kapcsolatban állnak más INGRES fejlesztő és döntést támogató eszközökkel, beleértve a képernyődefiniálást, lekérdezést, riport-generálást és grafikus eszközöket, stb.

V.szekció: U N I X ALAPÚ ALKALMAZÁSOK

GLOBUS INTEGRÁLT BANKI RENDSZER

LUKÁCSA LÁSZLÓ
KOPINT-DATORG RT.

A bankok és pénzügyi intézmények életében napjainkban bekövetkező drámai változások egyre inkább növelik az igényt a komplex információs rendszerek iránt. A következő oldalakon a GLOBUS integrált banki rendszerről található leírás mely a svájci COS Software Engineering AG.-nak a nemzetközi banki feladatok megoldására kidolgozott megoldása.

A COS AG magyarországi partnere a KOPINT-DATORG RT. részt vesz a terméknek a hazai viszonyoknak megfelelő formára hozásában és implementációjában.

Bevezetés

A GLOBUS egy új, teljesen integrált nemzetközi banki rendszer, mely lehetővé teszi bankoknak és más pénzügyi szervezeteknek, hogy megfeleljenek a gyorsan változó pénzpiaci területeken jelentkező kihívásoknak. A kereskedelmi, könyvelési, beszámolási és működési funkciók integrálásával egy teljes ügyfélkiszolgálói és háttérhivatali megoldását nyújt.

Háttér

A GLOBUS-t úgy fejlesztették, hogy egyedülálló, valós idejű támogatási környezetet teremtsen mind a nagy, a lakossági és a befektetési bankoknak, mind más pénzügyi szervezeteknek. A fejlesztő csoportnak az volt a szándéka, hogy figyelemre méltó banki tapasztalatukra támaszkodva egy olyan új, banki rendszert hozzanak létre, mely megfelel az 1990-es és azt követő évek üzleti követelményeinek. Az eredmény a GLOBUS, melyet már számos európai közép-keleti és afrikai állam pénzügyi intézeténél sikeresen bevezettek.

A megközelítés

A GLOBUS fejlesztése során a teljeskörű rugalmasságra való törekvés volt a jellemző. Ez a megközelítés egy olyan rendszert eredményezett, mely minden téren megfelel a felhasználók igényeinek. Az egyes módosítások végrehajtása az üzleti illetve működési paramétereket tartalmazó táblákban, segédprogramok segítségével történik, melyek lehetővé teszik a képernyők, lekérdezések és beszámolók megfelelő formára alakítását.

A segédprogramok valamint felhasználó által definiálható végrehajtási feltételek és paraméterek biztosítják a GLOBUS könnyű implementálhatóságát a legtöbb országban, úgy hogy semmi vagy csak minimális technikai segítségre van szükség. Ez a tulajdonság növeli a felhasználó önállóságát és mentesít a eladótól való függőség alól.

Mivel a GLOBUS moduláris rendszer, az egyes GLOBUS modulok felhasználásával egyedi, a felhasználó igényeit pontosan kielégítő megoldások építhetők. Ez tulajdonság teszi a GLOBUS-t rendkívül költségkímélővé, mivel a felhasználóknak csak a számukra szügséges modulokért kell fizetniük. Természetesen a kezdeti rendszerhez további modulok bármikor csatolhatók.

A GLOBUS a UNIX operációs rendszernek köszönhetően széles választási lehetőséget kínál a felhasználók számára a hardver eszközök kiválasztása terén, valamint jelentősen csökkenti mind a kezdeti beruházási, mind a folyamatos üzemeltetési költségeket is, melyek általában alacsonyabbak a UNIX alatt futó GLOBUS esetében, mint más egyedi rendszereknél.

A GLOBUS-t a szándékosan nem összegzett adatok jellemzik. Minden tranzakció vagy számlainformáció különállóan tárolódik, ahelyett, hogy a hagyományos rendszereknél megszokott módon főkönyvi számlákon összegződne. Az adatbázis felépítése ügyfél, termék, profit centrum orientált, így a felhasználó által megadott paraméterek alapján dinamikusan aggregált adatok nyerhetők. A lekérdezések és jelentések pontosságának biztosítása érdekében az adatok karbantartása a legalacsonyabb szinten történik Az összes lényeges adat on-line módon kerül frissítésre a valós idejű tranzakció feldolgozás során, igy biztosítva az információk elérhetőséget és abszolút pontosságát.

Kiemelkedő tulajdonságok:

A következőkben néhány olyan jellemző található mely megkülönbözteti a GLOBUS-t más hasonló banki rendszerektől:

- Nyilt rendszereknek megfelelő felépítés.
- A rendszer könnyen a valós élethez alakítható:
  - megadható a lekérdezések jelentések tartalma és formátuma
  - egyedi képernyők tervezhetők
  - feltételezett értékek határozhatóak meg
  - új mezők adhatók a képernyő és jelentés elrendezésekhez
  - a nyomtatott lapok formátuma megtervezhető
- Több fiók / több vállalat kezelés
- Több valuta kezelés
- Multi taszkos lehetőség WINDOWS környezetben.
- 12 nyelv használható egyidejűleg a különböző képernyők, jelentések és elektrónikus üzenetek szövegének a megjelenítése során, biztosítva mind a felhasználó, mind az ügyfél számára a választott nyelv használatát.
- A pénzügyi, mérleg és jegybanki jelentések dinamikusan állíthatók elő a számla és tranzakció adatokból.
- Teljes körű valós idejű áttekintés a kritikus kockázat kezelési elemekről, mint amilyen a hitel, kamat, valuta, likviditás, és kontingens érzékenység.
- Figyelemre méltó információ kezelő rendszer, mely valós időben nyújt GLOBUS és piaci információkat a kereskedelmi környezetben dolgozó dealerek és az intézet szintű áttekintést igénylő vezetők számára.
- Valós idejű tranzakció feldolgozás, elszámolás és kézbesítés.
- Egészen mező szintig lemenő, a teljes rendszert átfogó on-line segítség.
- Figyelemre méltó biztonsági rendszer, mely biztosítja a felhasználói tevékenység szabályozását egészen mező szintig.

- Teljes tranzakció "történet", melyből bármely végrehajtott tranzakció állapota látható.
- Teljesen autómatizált nap vége (end of day) feldolgozó és archiváló rendszer.
- A kiválasztott felhasználóknak számára közvetlen kapcsolódási lehetőség a biztosítása a GLOBUS rendszerhez, amin keresztül a számlájukal kapcsolatos információk és tranzakciók elérhetők.

A GLOBUS felépítése:

A GLOBUS számlázási, üzleti és tranzakció feldolgozó moduloknak egy funkcionálisan gazdag csomagja, mely egy központi mag és integrált adatbázis köré épül. Ezen túlmenően egy vezetői információs rendszer (MIS) működik együtt a GLOBUS-szal, újabb dimenziót nyitva a vezetői információk eléréséhez.

A mag (CORE SYSTEM)

A mag biztosítja az alapot melyre az egyedi üzleti igényeket legjobban kielégítő formára igazított rendszer felépíthető. A mag tartalmazza a többi GLOBUS modul számára szükséges funkciókat, mint amilyen a határértékek és kintlevőségek on-line karbantartása, biztonsági alkalmazások, jelentés és lekérdezési lehetőség, elektrónikus üzenet továbbítási lehetőség.

A sztandard mag a következő funkcionális részeket tartalmazza:

- Az adatbázis és adatrögzítési lehetőség
- Biztonsági rendszer
- Felhasználói segédprogramok
- Elektrónikus továbbító rendszer
- Ügyfél és belső számlák
- Mérleg
- Vezetői információ
- Kockázat kezelés

- Feldolgozási feltételek
- Auditálás
- Archiválási lehetőség

A sztandard maghoz kapcsolódva a következő opcionális modulok léteznek:

- Biztosíték (Collateral) modul
- Több-vállalatos feldolgozás
- Interface elektrónikus üzenettovábbító rendszerekhez
  - S.W.I.F.T.
  - Telex
  - CEDEL
  - Euroclear
  - Telekurs

Tanzakció feldolgozó modulok

Figyelemre méltó mennyiségű tranzakció feldolgozó modul áll rendelkezésre számos üzleti folyamat kezelésére:

- Kötvények
- Portfolió kezelés
- Pénzváltás
- Pénzpiaci müveletek
- Kereskedelmi hitelek
- Mortgage hitelek
- Letter of credits
- Lakossági szolgáltatások
  - Pénztár
  - Aláírás ellenőrzés
- Funds Transfer

Információ kezelő rendszer (IMS)

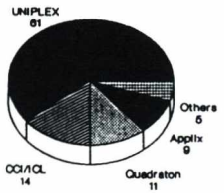
Az információ kezelő egy olyan rendszer, mely dinamikusan frissíti és megjeleníti a GLOBUS-on belüli illetve azon kivüli forrásból származó információkat. A valós idejű frissítés biztosítja a vezetők és dealerek számára a mindenkor naprakész, friss információkhoz való hozzáférést, azaz az IMS-t bármikor leolvasva az mindig az adott pillanatban meglévő helyzetet mutatja. Az IMS a következő lehetőségeket biztosítja:

- A GLOBUS, vagy más adatbázis pénzpiaci információs rendszeréből származó adatokat elérhetők.
- Bármilyen adaton, szabadon megadott képletek, formulák alapján történő számítások elvégezhetők.
- Valós időben frissíthetők a spreadsheet-ek és három dimenziós grafikák.
- A felhasználó igényeinek megfelelő képernyő formátumok határozhatók meg, mely bármilyen forrásból származó adatokat, számítási eredményeket és GLOBUS adat beviteli mezőket tartalmazhatnak.
- Felhasználónként 100 képernyő oldalnyi információ tárolásra alkalmas terület.

UNIPLEX iroda automatizálási rendszer
Taba Miklós KFKI-TRADIS Kft

A UNIPLEX iroda automatizálási rendszer alapja egy szövegszerkesztő, amit 1982-ben kezdtek el fejleszteni. Ujabb és újabb modulok hozzáadásával nőtt fel egy integrált, komplett rendszerré. 1992-re a világ legelterjetteb irodai software-e lett.

UNIX alapu irodai szoftverek, Europa



Source: Dataquest, 1992 May.

A következőkben egy rövid áttekintést kívánok nyújtani arról, hogy ezt az eredményt minek köszönheti, valamint arról, hogy maga a termék milyen elemekből építkezik.

Miért vált vonzzó termékké a UNIPLEX ?

A UNIPLEX integrált rendszer.

A UNIPLEX megoldást kinál egy átlagos irodában a következő felmerülő feladatokra:

- szövegszerkesztés
- dokumentumok nyilvántartása
- táblázatkezelés
- adatbáziskezelés
- űrlapok kitöltése
- határidőnapló használata

- terminus nyilvántartás, konferenciák szervezése
- telefon és címlisták
- üzenetek, dokumentumok továbbítása
- hozzáférési jogok kezelése

Az integráltság velejárója az egységes felhasználói felület. Aki megismerkedett a szövegszerkesztővel, az otthonosan fog mozogni a rendszer más területein is, mivel ott is hasonló vagy azonos menüpontokkal fog találkozni. Ezért a program kezelése igen gyorsan elsajátítható, ami tekintettel a programozásban nem jártas felhasznalókra alapvető szempont. Az integráltság további előnye, hogy a modulok "azonos nyelvet beszélnek", igen szoros kapcsolatban vannak egymással, így az adatcsere egyszerű és hatékony. Egy rövid példával megvilágitva: a táblázatkezelővel létrehozott üzleti grafika egy clipboard-on keresztül beleszerkeszthető egy szöveg file-ba. Hasonlóan egyszerűen lehet a táblázatkezelő egy mezőjében egy adatbázis meghatározott adataira hivatkozni. Lehetőség van a modulok közötti "közlekedésre", azaz az egyik modulban a felhasználó félbehagy egy tevékenységet, egy másikban elvégez valami más feladatot,és ennek az eredményét felhasználja az elsőben. Igy egyszerre lehet dolgozni pl. két vagy több szöveges dokumentumon, tábázaton, műveleteket végezhetünk a kalkulátorral, miközben állandóan a rendelkezésünkre áll a saját naptárunk.

A UNIPLEX nyitott rendszer

A UNIPLEX nyitott rendszer a hardware szempontjából, mivel gyakorlatilag korlátozás nélkül minden platformon működik ahol UNIX operációs rendszer létezik. Ez kb. 200 féle számítógép tipust jelent. Ha egy iroda olyan
mennyiségi növekedésen ment keresztül, amely indokolja a nagyobb teljesítményű számítógépre való átállást, akkor nem kell a megszokott software környezettől megválni, mivel a UNIPLEX támogatja a néhány PC-s rendszertől kezdve a mainframe-eket is (kb.1000 felhasználó).
A UNIPLEX nyitott rendszer a software szempontjából, mivel

- bármilyen UNIX program (saját fejlesztésű vagy sem) integrálható a rendszerbe. Az újonnan felvett modul szervesen beilleszthető a már meglevő menüpontok közé így biztosítva továbbra is az egységes felhasználói felületet.

- Lehetőség van valamely modul használatára egy saját fejlesztesű program keretein belül.
- Egyszerűen konfigurálható. Különböző felhasználóknak más és más feladataik, és hozzáférési jogaik lehetnek egy rendszeren belül. Ilyen esetben célszerű bizonyos menüpontokat felvenni vagy törölni, attól függően, hogy éppen ki használja. Igy ugyanaz a software képes kiszolgálni raktárost, titkárnőt és igazgatót is. Az ilyen tipusú konfiguráláshoz nem szükséges egyetlen programnyelv ismerete sem, mivel ez megoldható szöveg file-ok editálásával.
- Szöveges file-ok editálásával illeszthetők különböző terminálok és printerek is a rendszerhez. Ez a nagyfokú rugalmasság biztosít lehetőséget a rendszeradminisztrátornak, hogy bármilyen meglévő eszközt ki tudjanak használni a felhasználók.
- Egy rendszeren belül különböző nemzeti nyelvek használatára is lehetőség van (természetesn a hardware ill. a UNIX adta kereteken belül). Pl. egy levél megírásakor előfordulhatnak különböző nyelven írt részletek, és ezeket a részleteket a megfelelő nyelv szabalyai szerint lehet ellenőrizni a "spell-checker" segítségével, de az is lehetséges, hogy bizonyos felhasználó pl. angol nyelvű menüpontokkal találkozik, míg egy másik magyar nyelvűekkel és mindketten ugyanazokkal az adatokkal dolgoznak.
- A UNIPLEX támogatja a népszerű adatbázisok (INFORMIX, INGRES, ORACLE) használatát teljesen transzparens módon.
- A UNIPLEX rendelkezik egy saját fejlesztői nyelvvel (screen builder) mellyel gyorsan és hatékonyan lehet saját alkalmazásokat hozzászerkeszteni a kész modulokhoz. A programfejlesztéshez nem szükséges semmiféle compiler-t megvásárolni, hiszen itt is szöveges file-ok editálásáról van szó.
- A UNIPLEX lehetővé teszi grafikus felhasználói felület használatát .A "UNIPLEX Windows" X.desktop alkalmazás OSF/Motif Window Manager kontroll alatt működik.

A nyitott rendszernek köszönhetően a UNIPLEX tehát hardware szempontjából használható a PC-től a mainframe-ig, software szempontjából kibővíthető néhány soros programmal vagy akár egy teljes könyvelési rendszerrel és könnyű kezelhetőséget biztosít egy titkárnőnek és egy igazgatónak egyaránt.

A szövegszerkesztő

A szövegszerkesztő rendelkezik mindazon tulajdonságokkal amit a felhasználó elvár egy modern programtól:

- helyesírás ellenőrzés
- paragrafusok használata
- szöveg igazitások
- előre definiált vagy saját készitésű, egy vagy több oszlopos vonalzók (ruler-ek) használata
- fejlécek, lábjegyzetek használata
- oldalszámozás
- blokkok, file-ok összeszerkesztése
- keretek rajzolása
- szinoníma szótár
- glossary-k használata
- számolás
- megjegyzések hozzáfűzése
- könyvjelzők használata
- index létrehozás
- tartalomjegyzék készítés
- sortávolság beállítása
- ablakok használata
- különböző betűtipusok használata
- azonos levél küldése több személy számára

Lehetőség van az archivált levelek közötti keresésre tartalmuk, leírásuk, azonosítójuk szerint.

A táblázatkezelő

A UNIPLEX táblázatkezelője megfelel egy hagyományos táblázatkezelőnek (pl. Lotus 123) úgy használatában mint teljesítőképeségében.

A clipboard használatával a táblázat kiválasztott részeit mozgathatjuk, másolhatjuk. Nagyméretű táblázatok esetén ablakokat nyithatunk, így egyszerre vizsgálhatunk

különböző részleteket. Ha ezeket az ablakokat összekapcsoljuk akkor az aktív ablak mozgásával szinkronizált módon fog mozogni a többi ablak is.

A táblázat tetszőleges mezőjéhez hozzárendelhetünk adatbázis lekérdezést, így lehetőség van arra, hogy a tábázat módosítása nélkül mindig az aktuális adatbázis tartalomnak megfelelő grafikonokat, jelentéseket készítsünk.

Ha nem rendelkezünk nagy felbontású grafikus terminállal, akkor is generálhatunk grafikonokat melyek megjeleníthetők és nagy felbontásban kinyomtathatók.

A következő típusú műveletek végezhetők a táblázatkezelővel:

- statisztikai műveletek
- matematikai műveletek
- financiális műveletek
- szöveg manipuláló műveletek
- dátumon végzett műveletek
- logikai műveletek
- trigonometrikus műveletek

Táblázatokhoz hozzárendelhetők más (vele összefüggő) táblázatok, grafikonok, szöveges dokumentumok. Igy biztosított, hogy újraszámolás esetén mindig az aktuális eredményekhez jussunk.

A következő file formátumok használhatók:

- ASCII text
- Lotus 1-2-3
- DIF (Visicalc)

A UNIPLEX adatbáziskezelése

A UNIPLEX lehetőséget ad a tapasztalt, standard SQL-ben jártas felhasználoknak az adatbázisok lekérdezésére, módosítására. Parancs orientált módon, bármilyen az adatbázis által támogatott utasítasok kiadhatók, és elmenthetők későbbi lekérdezések céljából.

A UNIPLEX lehetőséget biztosít azonban programozásban semmiféle ismeretekkel nem rendelkezők számára is az adatbázis "formokon" keresztül. A felhasználó ebben az esetben is bevihet, módosíthat, törölhet, lekérdezhet adatokat de itt képernyő orientált módon, amely ember közeli kezelhetőséget biztosít.

A "DATALINK" modul hozzáférést biztosít INFORMIX, ORACLE, INGRES adatbázisokhoz.

Irodai segédeszközök

Elektronikus posta

Az elektronikus posta segítségével üzeneteket, utasításokat, dokumentumokat küldhetünk egyének ill. csoportok számára. A UNIPLEX belső nyilvántartását használva a tényleges személynevekkel dolgozhatunk, mely könnyebbséget biztosít a UNIX login nevekhez képest. Definiálhatunk csoportokat tevékenységi körüknek megfelelően és számukra egyszerre küldhetünk üzenetet.

Üzeneteket lehet titkosítani, kulcsszóval ellátni - ezzel is biztosítható, hogy csak az illetékesek olvashassák el.

Az üzenetekről másolatot küldhetünk más személyek számára, valamint hozzácsatolhatunk bármilyen dokumentumot pl. grafikát.

Az üzenetet küldhetjük "sürgős" megkülönböztetéssel, ekkor a címzett terminálján azonnal megjelenik egy figyelmeztetés és küldhetjük "ajánlva" azaz ellenőrizni akarjuk, hogy a címzett elolvasta-e küldeményünket.

Lehetőség van az üzenet láncolt továbbítására, azaz több személy számára küldjük. Ekkor ha valaki megkapta és esetleg saját megjegyzésével ellátta, akkor az üzenet továbbítódik a láncban előírt következő személyhez

A UNIPLEX naptár

A UNIPLEX segítségével időpontokat foglalhatunk le naptárunkban, kereshetünk bizonyos eseményeket megjegyzések vagy kulcsszavak szerint, esetleg egy bizonyos szöveg előfordulására vagyunk kíváncsiak.

Lehetőség van konferenciák szervezésére meghatározott személyek számára meghatározott helyen. Konferenciát csak olyan személy rendezhet akinek hozzáférési joga van az érintettek naptárjához, ui. a UNIPLEX automatikusan ellenőrzi a résztvevők és kijelölt terem foglaltságát. Amennyiben mindenki szabad, úgy bejegyzéseket tesz a résztvevők naptárjába, azaz a kérdéses időpontot lefoglalja. A szervező megjegyzésekkel utalhat a konferencia céljára.

A UNIPLEX kártya nyilvántartás

Egy helyen nyilvántarthatunk neveket, címeket, telefonszámokat, bármilyen számunkra fontos információt, melyek között később egyszerűen végezhetünk kereséseket. Vezethetünk mindenki számára hozzáférhető ill. csak saját használatra szánt nyilvántartást.
Ha a hardware támogatja a telefon használatát, akkor egy menüpont kiválasztásával automatikusan felhívhatjuk a kiválasztott személyt.

A UNIPLEX határidőnapló

Személyes használatra szolgál a határidőnapló, melyben teendőinket sorolhatjuk fel fontossági sorrendben, majd ezek között végezhetünk kereséseket különböző szempontok szerint.

A UNIPLEX űrlapkitöltő

Tetszőleges űrlapokat tervezhetünk, melyekre munkánk során szükség lehet, majd kitöltve vagy kitöltetlenül kinyomtathatjuk azokat.

Irodalom:

UNIPLEX felhasználói kézikönyvek

Korszerű Fulltext rendszer UNIX alatt

Herzceg István (ÁSzSz Rt.)- Mag. Peter Graf (PG u. P. KEG)

Mi a fulltext, illetve magyar nevén szöveges adatbáziskezelő?
Szabad szöveges gyors visszakeresést a teljes szövegből biztosító szoftvereszköz. Fulltexnek eddig nem a definiciónak helyesen megfelelő szoftvereket nevezték általában, hanem a különböző kulcsszavakra, vezérszavakra, deszkriptorokra épülő, struktúrálatlan (szabad) szövegek visszakeresését teszik többé-kevésbé könnyen lehetővé. De nem csak ilyenek vannak, hanem többféle módszertani megközelítés ismert ma már. Ezek alapjában a deszkriptoros módszer mellett a teljes tartalom szerinti keresés (search content) karakter összehasonlítással, illetve az invertált fájlos (szójegyzékes) módszer. Általában minden rendszer ezen alapelvek eltérő hangsúllyal vett keveréke.

A szöveges adatbáziskezelés eszméje több évtizede ismert, módszerei hosszú fejlődésen mentek keresztül. Időrendben a következő eljárások kerültek alkalmazásra:

- tezauruszos, vezérszavas, (deszkriptoros) módszer
- editor-jellegű, tartalom szerinti keresés (karakter-összehasonlító módszer)
- keresési lehetőség invertált listák (szójegyzék indexálás) szerint

A fenti három, elvileg különböző eljárásfajta előnyei-hátrányai a következők:

- **Deszkriptoros változat:** gyors keresés, de a deszkriptorozás időigényes, magasabb szaktudást igényel és gyakran kimaradnak lényeges vezérszavak.

- **Tartalom szerinti keresés karakter összehasonlítással:** minden szóra, szövegrészre kereshet, gyors a betöltés, de nagyobb állományoknál nagy a keresési időigény.

- **Invertált listás szövegindexelés:** előzőhöz képest több tárolási időt igényel, de rövid a keresési időigénye, bármilyen szövegösszefüggésre lehet keresni és a viszonylag kis indexállományok miatt viszonylag kicsi a tárolóigénye.

Részletezve tehát ez utóbbi módszer sajátosságait:

Az általunk fejlesztett szöveges adatbáziskezelő eszköz a numerikus adatbázistól abban különbözik, ahogy a dokumentumokat azonosítja. Így nem adott mezőket azonosít a rendszer, hanem az egyes szavakat. Ez bizonyos előnyökkel és hátrányokkal is jár.
Előnyök:

- **Gyors keresési idők:** Mivel az indexfájlok keresése többnyire binárisan történik, az indexelt szövegek nagysága nem játszik nagy szerepet. Ha pl a szöveg terjedelmét megduplázzuk, a kereséskor csak egy potlólagos fájlhozzáférésre van szükség.

- **Csekély helyszükséglet az adathordozón:** A tisztán szöveges adatbázisok nem rögített mezőhosszúsággal dolgoznak. A szövegek nem vesznek igénybe több helyet a szükségesnél. Az indexfájlok többnyire a szövegek 30-60 %-át igénylik, ami azonban a numerikus rendszereknél is gyakran így van, de azokkal ellentétben az állomány növekedésével az indexállomány fajlagos mértéke csökken.

Hátrány:

- Az indexek megszerkesztése - bár automatikus - időigényesebb a relációs adatbáziskezelőkénél.

A megvalósításra kerülő invertált listákkal dolgozó rendszer szolgáltatási a következők:

- 3-szintű felépítéssel rendelkezik, az ügyviteli sajátosságoknak megfelelően:
  adatbázis, dokumentum(irat), mező,
- ajánlott irányelvek
  * 1 dokumentum: felhasználó számára általában 3-5 oldal
  * legalább egy cím feltüntetése célszerű az első mezőben
- keresési opciók
  * szabad szövegű
  * mező szerinti
  * csonkoltra, hasonlóra keresés
  * logikai kapcsolatok (ÉS, VAGY, NEM, egyenlő, kisebb, nagyobb)

- szöveg input-output
  * Scanner-OCR
  * többféle szövegszerkeztő
  * adatbázis kezelők
  * táblázat kezelők

- saját szövegszerkesztő
- tetszés szerinti karakterkészlet
- irodai segéd funkciók
  * körlevél
  * formulák
  * etikett
  * lista készítés
- kommunikáció más szoftverekkel
- három védelmi funkció (adatbázis létesítés, ezen belül keresés, módosítás)
- kötőszavak letilthatósága
- deszkriptorok alkalmazása
- szinoníma szótár

A hatékony fulltext eszköz alkalmazási területei

A jelenlegi (ügy)iratkezelési gyakorlatban komoly gondokat okoz a régi ügyiratok visszakeresése. Ugyanis rendszerint egy ujabb ügynek az ügyorientált intézés következtében rendszerint van egy, vagy több elözménye, amelyre az ujabb ügy érdekében szükség van. Ekkor nagy nehézséget jelent, ha a hivatkozási azonosító (iktatószám, tárgyszó, stb.) nem ismert. Emiatt az ügyintézők idejének nagy hányada azzal telik - ahol még nem alkalmaznak korszerűbb eszközöket - hogy sok időt kell eltölteniük irattári kutatással.

Magyarországon sok ilyen eset adódik, mivel az iktatási rendszer következtében, ha egy ügyiratnál a keresés a tartalmi regiszterben (ez igaz a számítógépes iktatásra is) eredménytelen, annyi mintha az ügynek nem is lenne előzménye. Az előzmény reális feltárására a regiszterek (nyilvántartások) merev adatszerkezete (még ha az számítógépen is van) nem alkalmas, mivel azok egy esetnek legfeljebb a "csontvázáról" tudósítanak és végképp elveszhetnek azok az érdemi háttérkörülmények, amelyek egy ügy elbírálását emberközelivé teszik.

Kivezető utat az előzmény-háttér keresés problémáiból a szöveges adatbáziskezelő (fulltext) eszközök használata mutat. Éppen ezért a fejlettebb országok gépesített hivatalaiban az utobbi 1-2 évben robbanásszerűen terjednek ezek az eszközök. Ezek főbb jellemzője, hogy az ügyiratok - a külsők optikai karakterolvasón (OCR-en) keresztül, a belsők közvetlenül a szövegszerkesztőkből - bekerülnek a fulltext rendszerbe, ahonnan bármikor, bármilyen szó, szempont, fogalom, cím, vagy egyébb azonosító szerint a keresett ügyirat, akta, levél, jegyzőkönyv igen rövid idő alatt előhívható. Ha ilyen eszközöket - amelyek ára ma már elérhetővé teszi azokat - nálunk is kiterjedtebben alkalmaznának, feloldódna az a dilemma, miszerint, ha egy ügynél alapos előzménykutatást végeznek, azt nem lehet emberrel és idővel győzni, másrészt, ha gyorsítani kívánják az ügyintézést, számottevően csökkenhet a döntés megalapozottságának biztonsága.

A magyarországi hivatalok többségében eddig már bevezetett ügyiratkezelést korszerűsítő műveletek (szövegszerkesztés, sokszorosítás, telefax) után ma aktuálisan korszerűsíthető az irat (szöveg) kezelés is, mégpedig annak a következő három tényezője:

- tetszőleges szempontok szerinti irat-előkeresés
- egy szöveg tartalmi elemzése, ugyancsak tetszőleges szempontok szerint és
- az archiválás, helytakarékosan, jól hozzáférhetően.

Megoldás: korszerű fulltext (szöveges adatbázis kezelő) eszköz, ahol arra a kérdésre, hogy mire használható? Alapvetően a következő válasz adható: minden szöveges iratot "letárol", majd abból bármilyen összefüggésben, kényelmesen, gyorsan, bármit elő lehet keresni. (Például irattárban, jogszabály gyűjteményben.)

A fulltext rendszerek felhasználási köre nem csupán jogszabály-visszakereső, vagy ügyirat kezelő funkcióban található, hanem sok, más alkalmazási területe is van. Jelenleg külföldön főleg irodalom-figyelő adatbázisokhoz, ügyfélnyilvántartásokhoz, ill. nagy adatbázisokból lekérdezett blokkokban való kutatásra használják. Használatos még a betegellátásban, pl a betegekről készült zárójelentésekből különféle elemzések készítésére, továbbá precedens-esetek kutatására döntési eljárásokhoz, biztosítási kárrendezésekhez. Ez utóbbi felhasználási kör bizonyos analógiát mutat a bűnesetek felderítését, továbbá minősítését szolgáló tevékenységekkel. Mindezeknek a támogatása fulltext adatbázissal úgy látható el, hogy a különböző eseteket leíró anyagokat fulltext adatbázisban letárolják és ebből az adatbázisokból elemzéseket végeznek. Ilyen elemzés lehet pl:

- hasonló, ill. egyező jegyek, jellemzők vizsgálata egy aktuális eset és más korábbi esetek között,
- összefüggés keresése több aktuális eset között hasonló, ill. azonos jegyek (jellemzők) szerint,

- aktuális esetben ítéletalkotás támogatása korábbi, hasonló esetekben megtett ítéletalkotások kigyűjtésével,
- személyleírás összevetése korábbi személyleírásokkal.

Magyarországon eddig a szöveges adatbázis-kezelés tezauruszos (vezérszó orinentált) módszere terjedt el. A jelenleg használt szöveges adatbáziskezelők közös jellemzője, hogy csak a tartalom fő adatait és egy hiearchikus tezaurusz rendszert tartalmaznak, általában fejezet, tétel bontásban. Gondot okoz ezeknél a rendszereknél, hogy egy dokumentum megjelenése és deszkriptorozása között nagyobb átfutási idő szükséges. A kapott szöveget invertált lista típusú szöveges adatbáziskezelőbe beolvasva, azonnal egy bármilyen szempont, fogalom, szó-összetétel alapján kereshető teljes szövegállományhoz jutnak.

Fulltext rendszer kapcsolódása a UNIX operációs rendszerhez.

A felhasználói igények növekedése, az osztott adatbázis elérésének szükségessége, valamint a teljesítmény iránti igények a magasabb kategóriájú rendszerek felé irányították a figyelmet. Ezeknek a kritériumoknak optimálisan leginkább a nyilt rendszer(ek) felelnek meg, aminek jelenleg legjellemzőbb képviselője a UNIX. A UNIX eredetileg szövegfeldolgozásra készült, amit alátámaszt az, hogy a legtöbb szabványos eszköz szöveges jellegű fájlok kezelésére íródott. Ilyen rendszerbe kitünően integrálható egy fulltext eszköz.
A programnak két megjelenési formáját terveztük. Az első az interaktív, párbeszédes mód, ahol a felhasználó tetszőleges műveletet elvégezhet. Az interaktív megjelenést egy opcionális Xwindows interfész segíti, amely egyben a program hordozhatóságát is javítja.

A második megjelenési formája a parancsmódú használat. Ebben a módban lehetőség van interaktív módban végrehajtott és letárolt makroformájú keresések, illetve makronyelven definiált keresések végrehajtására. Jól használható parancsláncba szervezve a meglévő eszközök és lehetőségek felhasználásával. (**sed, awk, uniq, troff,** input-output átirányítás, csövek). Ezek a lehetőségek komplex keresési stratégiák kialakítására alkalmasak. A rendszerhez való kapcsolódást program által generált kompatibilis fájlformátumok segítik. Implementálása a szabványokhoz igazodva történik, s így hordozhatósága a kismértékében eltérő UNIX rendszerek között nem okoz problémát.

Az irodai automatizálás problémái
A PRISMA Office,
mint megoldás

Palkó Gábor
ONYX Szoftverház Kft.

Hosszú évek fejlesztői-üzemeltetői tapasztalatai egyre inkább világossá tették számunkra, hogy nincs az az iroda, amelyik a számítástechnika *"betörésével"*, ne igényelne valamilyen kifejezetten *"csak"* az irodai munkát támogató rendszert. Talán mások számára is ismerős ez a jelenség, ami a számítógépes rendszerek telepítése során szinte mindenhol tapasztalható: először csak az un. számításigényes, nagytömegű adatokat mozgató és előállító rendszereket keresik a felhasználók. Később azonban ezek megszokása és betanulása után szinte maguktól kezdenek olyan igényekkel előállni, hogy talán jó lenne az a számítógép másra is. Mondjuk szövegek, levelek szerkesztésére, különböző dokumentumok előállítására. Ez az igény gyorsan kielégíthető, hiszen a piacon sok remek szövegszerkesztő program kapható (másolható).

Ez az a pont, amikor egy ilyen irodában már biztos, hogy előbb vagy utóbb, de keresni fognak *"valami jobb"* rendszert. Azt, hogy miben jobb az áhított rendszer, a felhasználó ekkor még nem is tudja igazán megfogalmazni, csak érzi.

Hogy miből érzi?

- Kis idő elteltével már remek leveleket, dokumentumokat szerkeszt ő is és a kollégák is. Csak az a bökkenő, hogy nem azonos programot használnak, így nehézkes a közös munkavégzés.

  "Kéne valami egységesítés"

- A meglévő rendszerek adatait igen sokszor fel lehetne használni a levelek, dokumentumok írása során. Adatok leválogatása, kiírása (ASCII), import, átszerkesztés, ... nehézkes, de megy.

  "Azért jó lenne egy kicsit egyszerűbb adatcsere"

- Egyre több különböző rendszert használnak az irodában. Fáradtságos ezek külön indítása, kezelése, a rendszerek közötti mozgás.

"Jó lenne ezeket integrálni valamibe"

- "Papírmentes iroda", mindent a számítógéppel! Szerződések, számlák, levelek, kimutatások ...

"Tárolás, visszakeresés, archiválás!!!"

- Az egyre több dokumentum különböző kezelési megoldásokat igényel, nem elegendő csak eldugni a rendszerben az egyes védett anyagokat.

"Védelem, titkosítás, jogosultságok"

- És mivel minden felhasználó étvágya a már *"elfogyasztott"* rendszerek mennyiségével arányosan nő, így a további igények lavinaként zúdulnak a szakemberre.

"Kommunikáció"
"Elektronikus levelezés"
"Határidő napló"
"Táblázat kezelés"
"Faxolási lehetőség"
"Kiadványszerkesztés
"..."

A szakember ekkor már csak egyet tehet, szétnéz a piacon, hogy mit lehet kapni. Melyik az a rendszer, ami megközelíti a felhasználó igényeit és egyben eleget tesz az általa egy ilyen rendszerrel támasztott feltételeknek is.

Hogy mik ezek a feltételek?

- Legyen a rendszer magyar, de ízig-vérig! (Menük, parancsok, helpek, leírás, billentyű, képernyő és nyomtató kezelés).

- Biztosítsa a rendszer a gyors betanulást, könnyen lehessen kezelni. Ne kelljen két diploma a használatához!

- Könnyen *"idomuljon"* az új rendszer az irodához. Az átállás, a meglévő rendszerek, adatok integrálása a lehető legkevesebb energiát igényelje.

- A rendszer a lehető legkevesebb *"szakmai"* beavatkozással legyen üzemeltethető.

- Az esetleges újabb, ma még nem látható igényekkel szemben is nyitott legyen.

- Tudjon *"mindent"*, de ezt ne zúdítsa egyszerre a felhasználóra. Támogassa a fokozatos betanulást, kezdve az egyszerű írógép kiváltásával egészen a kiadványszerkesztés lehetőségéig.

- Platform függetlenség. Ne kelljen egy új rendszer után nézni, ha az iroda kinövi a jelenlegi hardvert.

Létezik már ilyen rendszer?

Természetesen! Ma már sok, a fent felsorolt tulajdonságok szinte mindegyikével rendelkező irodai rendszer kapható a piacon. Csakhogy mindegyikből hiányzik valami. Az esetleges kompromisszum meghozatala előtt jó, ha a szakember alaposan tanulmányozza a szóba jöhető rendszereket. Mi az, aminek a hiánya még elviselhető a számára, és mi az amiről végképp nem tud lemondani? Ez egy igen nehéz feladat. Ebben kívánunk most segíteni, bemutatva egyet (egyetlent?) a lehető legjobb megoldások közül.

Ez a PRISMA Office Iroda Automatizálási rendszer, mely tartalmaz:

- egy könnyen és igen gyorsan megtanulható szövegszerkesztőt, mely magyarul beszél (parancsok, menük, makró nyelv), a képernyőn és a nyomtatókon is helyesen kezeli a magyar karaktereket. A rendszer teljes magyar dokumentációval rendelkezik. Az egy rendszerben dolgozók különböző nyelven használhatják a programot (angol, német, francia, holland, magyar).

- *elektronikus üzenetközvetítőt*, mely lehetővé teszi a felhasználók, felhasználó csoportok számára a gyors és dokumentálható információ küldést (dokumentumok, adatok, üzenetek, ...).

- *határidő naplót*, melyben egyedi és csoportos feladatok elvégzésére figyelmeztető *"emlékeztetőket"* lehet elhelyezni, melyek hatására a rendszer az esemény bekövetkeztére automatikusan figyelmeztet a beállított gyakoriságnak megfelelően.

- a dokumentum könyvtár és a *dosszié* kezelés révén a rendszer dokumentumai hierarchikus formában tárolhatók (az összetartozó dokumentumok azonos dossziéban). Ennek segítségével a dokumentumok visszakeresése, kezelése gyors és át-

tekinthető. A dokumentum és dosszié nevek tetszőleges, max. 35 karakterből álló nevek lehetnek. Ezzel a nagytömegű adatok esetén is teljes biztonsággal kereshetők vissza a rendszer adatai.

- különböző *biztonsági*, illetve elérési *jogosultsági szintek* definiálhatók az egyedi felhasználók és felhasználói csoportok szintjén.

- az integrált *adatbázis kezelés* segítségével különböző, akár más rendszerek adatai is gyorsan integrálhatók a dokumentumokba. (Címlisták, kapcsolatok, ...)

- dokumentum archiválási és visszakeresési lehetőségek.

- integrált *fax-kezelési* lehetőségek. Faxok automatikus küldése, fogadása, a forgalom automatikus regisztrálásával.

- *táblázatkezelés*, Lotus, Symphony, Excel táblázatok dinamikus integrálása a szövegbe.

- grafikák integrálása szövegbe, az összes elterjedt formátum kezelésével (PCX, TIFF, WPG, ...).

- kezeli a különböző szövegszerkesztő formátumokat (Word, WordPerfect, ...)

- beépített *soros kommunikáció*, melynek segítségével más rendszerekbe csatlakozva (MAILBOX, ...) adatok gyors cseréje valósítható meg.

A rendszer hardver igénye:

- A rendszer fut DOS, OS/2 operációs rendszer alatt az összes ismert hálózati rendszeren IBM PC kompatibilis számítógépeken, valamint különböző UNIX operációs rendszerek (SCO UNIX, SCO XENIX, HP-UX, AIX, SINIX, ...) alatt IBM RS/6000, Aviion, HP 9000, Altos 1000, Siemens, Unisys, SUN, ... gépeken.

- A rendszernek létezik WINDOWS alatt futó változata is, melynek funkciói megegyeznek a karakteres változat lehetőségeivel, de néhány funkcióban a WINDOWS adta lehetőségek (DDE, OLE) kihasználásával ez a változat további szolgáltatásokat nyújt.

VI.szekció: **A 90-es ÉVEK STRATÉGIÁI**

A Digital nyílt rendszer stratégiája a 90-es évekre

Török Bálint, Digital Equipment Hungary Ltd.

1) Bevezetés

Napjainkban a számítástechnika felhasználóinak számos kihívással kell szembenézniük. A piacért folyó kiélezett versengés megkívánja, hogy egy vállalat termékei a lehető leggyorsabban piacra kerüljenek, csak így szerezhető és tartható meg a versenyképesség. Ugyanakkor a termékek és szolgáltatások minőségét folyamatosan javítani kell. Mindezeket az üzleti célokat csak olyan környezetben lehet elérni, amely lehetővé teszi a termelékenység növelését, a költségek csökkentését. Hogyan érhetők el ezek a célok, oldhatók meg a problémák a mai gazdaságban? A vállalat minden alkalmazottjának időben meg kell kapnia a megfelelő információt, hogy a probléma megoldásához szükséges döntést meghozhassa.

Ez az információigény számos kérdést vet fel. A felhasználó a piacon lévő legjobb megoldást akarja megvenni, anélkül, hogy bármelyik gyártó (hardware vagy software) mellett elkötelezné magát, de a megvásárolt elemeknek együtt kell működniük . A megoldás teljes megvalósítása időbe telik, de a legrövidebb időn belül el kell kezdeni azt használni, mindemellett felmerül az igény, hogy a jövőben megjelenő új technológiák a rendszerbe beintegrálhatóak legyenek. A felhasználó a nyílt rendszerek felé akar közeledni, de meg akarja őrizni eddigi hardware, software befektetéseit is, használni akarja a felhalmozott szakértelmet. Mindemellett költségeit is egy bizonyos szint alatt kell tartania. Hogyan lehetséges mindezt megoldani?

Ezen előadás célja, hogy választ adjon ezekre a kérdésekre a Digital nyílt rendszer stratégiájának ismertetésével.

2) A nyílt rendszerek definíciója

A "nyílt rendszerek" kifejezést sokan - gyártók, a sajtó, különféle szervezetek - sokféleképpen használták, használják, ami kisebbfajta zűrzavart eredményezett a fogalom értelmezésében. Sokan a nyílt rendszereket azonosítják egy bizonyos operációs rendszerrel, egy bizonyos hardware platformmal, vagy a nagy tömegben eladott rendszerekkel, holott itt sokkal többről van szó, egy új stílusú számítástechnikáról.

A Digital nyílt rendszer definíciója a következő. A nyílt rendszer gyártó semleges számítástechnikai alkalmazás környezet, amely ipari szabvány interface-ekre épül. A nyílt rendszer szabványos interface-ei biztosítják az alkalmazói programok, adatok, információ hordozhatóságát, együttműködését. A nyílt rendszernek együtt kell tudni működnie a már meglévő számítástechnikai környezettel. Tehát a Digital nézete szerint a nyílt rendszerek építőelemei az ipari szabványoknak megfelelő interface-ek, nem pedig néhány technológia szűk készlete.

3) A Digital szerepe a szabványosításban

A Digital már bebizonyította hosszú távú, erős elkötelezettségét a szabványok implementálása iránt. A szabványok voltak a VAX/VMS rendszer erősségei a múltban és ma is, melyek most egészültek ki olyan nyílt rendszer interface-ekkel mint a POSIX, X/Open XPG3. A VMS volt az első nem UNIX operációs rendszer amely X-window felületet nyújtott, amely ma már az OSF-Motifot is támogatja. Az ULTRIX - a Digital UNIX implementációja - volt az első X-window-t és POSIX interface-t nyújtó rendszer, az elsők között felelt meg az X/Open XPG3 előírásainak. A Digital legjobb technológiáit kínálta fel a nyílt rendszereknek (pl. X User Interface - XUI), az OSF DCE 7-féle technológiájából 4 Digital fejlesztés. A Digital szakemberei tagjai a szabványokat megalkotó

testületeknek, jelen vannak, és aktívan közreműködnek az új szabványok kidolgozásánál.

4) A Digital nyílt rendszer stratégiája

A nyílt rendszer stratégia alapja egy fejlett architektúrájú nyílt alkalmazás programozási interface biztosítása az összes Digital platformon (Alpha, VAX, MIPS, Intel). Ezen túl a Digital fejleszt alkalmazásokat más gyártók platformjára, szolgáltatásokat nyújt saját és más UNIX platformokra, alkalmazásokra, továbbá biztosítja az együttműködést a már meglévő számítástechnikai környezettel a NAS (Network Application Services) termékcsaláddal. A NAS egy ipari szabvány interface-eken alapuló middleware eszközkészlet, melyet a VMS, az ULTRIX (UNIX), az OS/2, DOS, MAC, SNA platformok közötti kliens-szerver alapú együttműködés megvalósítására definiált a Digital. A NAS által megvalósul a valódi multi-vendor kliens-szerver számítástechnikai környezet, melynek elemei egymással összekapcsolódnak, együttműködnek, adatot osztanak meg, stb.

A platform stratégia 3 fő operációs rendszere a Windows/New Technology (NT), az OSF/1 UNIX, és az OpenVMS. Az NT három hardware platformon fog futni: Alphán, MIPS RISC-en, és Intel alapú gépeken. Az OSF/1 ugyanezen a három platformon fog futni, míg az OpenVMS az Alpha és VAX hardware platformot támogatja. Tehát mindhárom operációs rendszer többféle hardware-t támogat, ezáltal megfelelnek a különböző ár/teljesítmény és felhasználói igényeknek.

5) A Digital UNIX stratégiája

A Digital UNIX stratégiája egyértelműen az OSF/1-en alapul. Az OSF/1 technológia biztosítja az egyik legmodernebb UNIX környezetet, és átmenetet biztosít a mai ULTRIX-tól a

MIPS-en és Alphán futó OSF/1-hez. ULTRIX-on több mint 3000 alkalmazás fut, és a software házak már aktívan dolgoznak alkalmazásaik OSF/1 verzióin.

A két rendszer egymás mellett élése, illetve az átmenet (migráció) igen fontos kérdés. A mai ULTRIX-ot futtató hardware-en (MIPS), fut, futni fog az OSF/1. Az OSF/1-et úgy tervezték, hogy a lehető legnagyobb mértékben kompatibilis legyen az ULTRIX-szal, így az ULTRIX alkalmazások jelentős része újrafordítás nélkül is fut az OSF/1-en. A jövőben mind MIPS-en, mind Alphán az OSF/1 ugyanazon verziója fog futni , megkönnyítve a MIPS-Alpha átmenetet.

A jelentősebb UNIX szállítók mindegyike rövidesen verziót vált. A Digital teljes mértékben elkötelezte magát az OSF irányában. A Sun kivételével az összes jelentős gyártó bejelentette, hogy részben vagy egészben, implementálja az OSF/1-et. A Digital DEC OSF/1 v1.0 operációs rendszere az év elején jelent meg a piacon, mint a világ első OSF/1 implementációja. Az IBM és a Hewlett-Packard is az OSF/1 technológiát választotta, a UNIX System Laboratories be fogja integrálni rendszerébe az OSF AES-t (API), a SCO is dolgozik már saját OSF/1 implementációján.

6) Az OSF/1 operációs rendszer

Az OSF/1-et az Open Software Foundation szervezet dolgozta ki. Az OSF gondosan kiválogatja az iparban megtalálható legjobb technológiákat, ugyanakkor nagyon ügyel arra, hogy az összes jelentős ipari szabványnak megfeleljen az OSF/1 rendszer. Az OSF/1, míg magában foglalja a hagyományos UNIX tulajdonságokat, napjaink egyik legmodernebb UNIX rendszere. Az alaprendszert modern tervezési módszerek felhasználásával újraírták, ilyen pl. a Mach kernel, melyet eredetileg a Carnegie Mellon egyetemen fejlesztettek ki. Az OSF/1 Mach alapú mikro-kernelt tartalmaz.

Az operációs rendszer szintjén túl az OSF számos eszközt

nyújt. Ilyen pl. az X11 alapú OSF/Motif user interface. A DCE (Distributed Computing Environment), DME (Distributed Management Environment) fejlesztése jelenleg folyik, ezek az eszközök további felhasználói igényeknek fognak megfelelni. Az OSF szervezet biztosítja az implementációk ellenőrzését, ezen a DEC OSF/1 v1.0 sikeresen átment.

Bár az OSF/1 piac ma még kicsi, de mivel sok gyártó támogatja, a holnap UNIX rendszere lesz. Független forrás szerint (Info Corp.) 1992-ben az OSF/1 piac még nem éri el az 1 milliárd dollárt, 1993-ra ez már 4-5, 1994-ben 7-8, 1995-ben 13-14, 2001-ben 35 milliárd dolláros lehet az OSF/1 piac.

A Digital jelenlegi UNIX kínálatát - ULTRIX, VAX System V, OSF/1 v1.0 - egyetlen termékben fogja egyesíteni, és ez az OSF/1 egy későbbi változata lesz. A mai DEC OSF/1 v1.0 elsősorban fejlesztésre és munkaállomás felhasználóknak készült. A következő verzió, a v1.2 már mindkét - Alpha és MIPS - platformra kapható lesz, és ez már bármilyen - pl. kereskedelmi - célra jól használható lesz. A DEC OSF/1 System V kompatibilis (SVID 3) lesz a jövőben, a SVID 3 alkalmazói interface (API) implementálása révén, továbbá magasfokú bináris és forrás szintű kompatibilitást fog biztosítani az ULTRIX-szal.

Röviden tekintsük át a DEC OSF/1 néhány jellemzőjét. A hagyományos UNIX jegyek közül megtaláljuk a BSD-t, SVID2 kompatibilitást, 3 féle shellt: C, Bourne, Korn, ANSI C, System V és BSD file systemet, az NFS-t. A DEC OSF/1 a következő főbb szabványoknak felel meg: POSIX 1003.1, X/Open XPG3, OSF AES, SVID2, X-window system, OSF/Motif. A Mach kernel a v1.0-ban még nem implementálja a mikrokernel architektúrát, ezt a következő verziótól támogatja a DEC OSF/1, ugyanúgy a szimmetrikus multiprocesszálást is. Az ULTRIX még nem teszi lehetővé az osztott könyvtár használatot, a DEC OSF/1 már igen. Tartalmaz egy logikai kötet kezelőt - ennek segítségével pl. egy nagy file több disken tárolható -, disk tükrözési lehetőséget. A real-time jellemzők közé tartozik a POSIX 1003.4-nek való megfelelés, továbbá a preemptív kernel. Számos a produktivitást támogató window alapú alkalmazás is segíti a felhasználót.

Ez év elején, a DEC OSF/1 megjelenésekor már kb. 100 alkalmazás támogatta az új operációs rendszert, ma ez a szám jóval meghaladja az 500-at, ezek között szerepelnek különféle CASE eszközök, compilerek, adatbáziskezelők, stb. Néhány fontosabb példa: Acucobol, Autodesk, Cadre, ESRI, Informix, Ingres, Oracle, Progress, Interleaf, Unidata.

Az alkalmazások hordozhatósága és együttműködésük biztosítása érdekében egy ipari szabványokon alapuló alkalmazási környezetet kellett létrehozni. Ennek szemléltetésére a Digital az ún. hexagon modellt használja. Képzeljünk el egy hatszöget, melynek közzéppontjában az alkalmazás van, környezete pedig 6 önálló részből áll, ezek: operációs rendszer, programozási nyelvek, felhasználói interface, hálózati interface, adatbázis kezelés, elektronikus dokumentum kezelés. Tekintsük végig, hogy a DEC OSF/1 esetében az alkalmazás környezetének hat eleme milyen szabványoknak felel meg. Mint láttuk, az operációs rendszer oldaláról ezek az X/Open XPG3, OSF AES, SVID, POSIX 1003.1, 1003.4. A programozási nyelvek ANSI nyelvek: C, C++, Fortran, Pascal. A felhasználói és grafikai interface az X11 alapú OSF/Motif, a GKS, PHIGS, PEX. A hálózati oldal elemei: TCP/IP, XTI, BSD 4.3, DCE. Az adatbáziskezelés az SQL-re épül. A dokumentum kezelés szabványai pedig a CDA, és a Postscript.

A hexagon modell is megmutatja, hogy egy nyílt rendszer messze többet jelent az operációs rendszernél.

7) OpenVMS

Az OpenVMS stratégiának három eleme van.

Az első a platform nyíltsága. Az OpenVMS minden olyan jellemzővel bír, mint bármely más modern nyílt rendszer: OSF/Motif felhasználói interface, szinte az összes ANSI programozási nyelv, szabványos grafika (GKS, PHIGS), szabványos adatkezelés (EDI X.12, SQL), POSIX és X/Open XPG3 kompatibilitás, OSI és TCP/IP hálózati protokoll, továbbá Ethernet, Token Ring, FDDI támogatás. A VAX 6000 rendszer

támogatja a VME buszt.

A második elem a versenyképesség, az ár/teljesítmény arány. A mai VAX-ok bármelyik modern RISC rendszerrel állják az összehasonlítást, a TPC-A teszt alapján pedig a piac legnagyobb teljesítményű gépei között szerepelnek.

A harmadik stratégiai elem az OpenVMS gazdag funkcionalitása, minősége és különösen olyan kereskedelmi jellemzői, mint az elosztott tranzakció feldolgozás, kliens-szerver architektúra, cluster technológia, adatbiztonság, flexibilitás.

A fentiek közül vizsgáljunk meg egy olyan aspektust, amely a UNIX felhasználók számára fontos lehet, ez pedig a POSIX és X/Open XPG3 kompatibilitás. Mit jelent mindez? Azt jelenti, hogy amennyiben egy alkalmazást úgy fejlesztünk az OpenVMS környezetben, hogy az szigorúan megfeleljen az XPG3, és POSIX előírásoknak, akkor ezen alkalmazás forrását átvive egy másik XPG3 és POSIX konform platformra - pl. OSF/1, ULTRIX, SCO UNIX, stb. - az módosítás nélkül újrafordítva az új platformon is működni fog! És természetesen ez fordítva is igaz.

A felhasználó UNIX stílusú parancsokkal, shell-ekkel, segédprogramokkal dolgozhat (standard UNIX parancsok, rendszerhívások, C fejlesztői eszközök, c89, cc, yacc, lex, lp, lpstat, ctags, make, egrep, stb.)

8) Hardware platformok

Legelsőnek említsük meg az Intel x86 platformot, ezen jelenleg az SCO UNIX fut, az SCO az OSF/1 irányt választotta.

A jelenleg kereskedelemi forgalomban lévő Digital RISC gépek a MIPS R3000 RISC processzorára épülnek, az ULTRIX - amely lényegében BSD 4.2, 4.3 UNIX -, és az OSF/1 operációs rendszert futtatják. Mindegyik MIPS RISC gép processzor kártyája cserélhető, akár egy gyorsabb R3000-re, akár a közeljövőben megjelenő R4000 CPU upgrade kártyára.

Szintén a közeljövőben jelenik meg az OSF/1-et és az

OpenVMS-t támogató teljes Alpha család, melynek lesznek desktop, deskside, és datacenter modelljei. Röviden néhány szót az Alpháról: az Alpha 64-bites RISC processzor, a Digital saját fejlesztése, a chip első generációjának maximális órajel frekvenciája 200-MHz, amivel 400 MIPS és 200 MFLOP csúcsteljesítményre képes. A low-end Alphákon az NT is futni fog. Az Alphán az OSF/1 valódi 64-bites változata fog működni. Idővel az Alpha lesz a Digital UNIX stratégiájának elsődleges platformja.

Míg végül, de korántsem utolsósorban tekintsük a teljes VAX családot. Ezeken a gépeken az OpenVMS/VAX fut. A nemrégiben megjelent VAX-7000 és VAX-10000 gépcsalád már készen áll az Alpha technológia fogadására, egyszerűen a CPU kártya cseréjével Alphává upgrade-elhető. Az alkalmazott busz az ipari szabvány FutureBus+.

Az IBM és a nyílt rendszerek

Szabó Balázs - K.Szabó Zoltán IBM Magyarországi Kft

Az IBM neve végigkíséri a számítástechnika fejlődését. Ezen írásunkban azt szeretnénk érzékeltetni, hogy a napjainkban egyre inkább tért hódító nyílt rendszerű architektúrák által támasztott követelményeknek hogyan felelt meg ez a vezető amerikai cég. A következőkben először rövid áttekintést adunk a nyílt rendszerek (leginkább a UNIX) IBM-en belüli evolúciójáról. Ezt követően bemutatjuk a "zászlóshajót", az IBM RISC/6000-es számítógépet. Ki fogunk térni néhány alkalmazásra és tárgyaljuk a különböző számítógéptípusok közötti kommunikációs lehetőségeket is.

Az AIX operációs rendszer kialakulása

A UNIX-nak az 1960-as évek végétől növekvő népszerűsége a számítógépgyártókat egyre inkább arra sarkallta, hogy ezt az operációs rendszert is ajánlják a lehetséges megoldások között. Az IBM 1979-ben kezdett UNIX-fejlesztésbe, amit az ATT Bell Laboratóriumával közösen végzett. A cél egy S/370-es gépen futó UNIX-rendszer volt. 1983-ból származik a Series/1 modellen futó UNIX-változat. Az IBM PC elterjedésével egy korai, IBM PC/IX nevű UNIX-like operációs rendszer megszületése is együttjárt, amit a "Big Blue" 1984-ben jelentett be. Ugyanekkortájt jelent meg az S/370-es mainframe UNIX operációs rendszere is. Meg kell azonban jegyeznünk, hogy az egyéb operációs rendszerek (géptípustól függően a DOS, VM vagy az MVS) nagy sikere, a rajtuk megírt alkalmazások száma nem ösztönözte eléggé az IBM-et a UNIX-alapú gépek erőteljesebb fejlesztésére. A UNIX az üzleti alkalmazásokban (adatbáziskezelésben, hozzáférési jogok magadásában, adatbiztonság, stb) eleinte nem elégített ki olyan követelményeket, amelyeket más rendszerekben megvalósítottak.

A RISC-technológia kialakításával 1986-ban az IBM piacra hozta RT PC nevű számítógépét, amit elsősorban ipari alkalmazások futtatására tervezett. Az RT előtag a RISC Technology névre utal. Ez a számítógép volt az első, amelyik csökkentett utasításkészletű processzorral dolgozott. A gépen egy újabb, UNIX-szerű operációs rendszer futott, ami az Advanced Interactive Executive (AIX) nevet kapta. A hardver elsősorban az USA-ban terjedt el, az operációs rendszer azonban jó alapját képezte az 1990-ben megszületett általánosabb célú RISC/6000-es gépcsaládnak.

Tekintsük most át azt, hogy mit nevezhetünk nyílt rendszernek és az AIX operációs rendszer mennyiben elégíti ki a követelményeket.

Az AIX mint nyílt rendszer

A UNIX-ot szívesen emlegetik nyílt rendszernek, ami mögött azonban talán túl sok különböző elgondolás húzódhat meg. Nyílt rendszerűnek nevezhetjük, mert a kezdeti időkben az operációs rendszer forráskódjával együtt mindenki számára elérhető volt, ami nagyban segítette megértését és elterjedését. Ennek oka az volt, hogy az "ősi" verziót kifejlesztő ATT Bell Laboratóriuma nem kezelte a programrendszert üzleti vállalkozásként.

A UNIX manapság azonban egyre inkább jó üzlet, a forráskód immár nem része az operációs rendszernek, esetleg külön megvásárolható a szállítótól (pl. az AIX 3.1. az IBM-től). Az elmúlt mintegy tíz évben a számítógépet vásárlók köre eljutott egy olyan fokra, amikor pontosan meg tudja nevezni egy géptípussal szemben az igényeit. A felhasználók jogot formálnak arra, hogy ne legyenek kiszolgáltatott helyzetben, és egyre nagyobb alkalmazási programjaikat, adatbázisaikat könnyen tudják más géptípusra átültetni. Ez a kívánság gyors ütemű szabványosítást eredményezett és eredményez, amelyet manapság minden számítógépgyártó elfogadott. Kétségtelen az is, hogy a nyílt rendszerű architektúrák alatt elsősorban a UNIX-like operációs rendszereket és a hozzájuk kapcsolódó alkalmazásokat értjük. Az IBM a nyílt rendszerekkel kapcsolatos elvárásait a következőképpen foglalja össze (mindez összhangban van az Open Systems Foundation megfogalmazásával):

A nyílt rendszerű környezetre olyan, nemzetközileg elfogadott szabványok alkalmazása jellemző, melyek interfészeket, szolgáltatásokat és alkalmazásokat definiálnak, ezáltal biztosítva az egyébként különböző architektúrák között a magas szintű kommunikációt , a felhasználói programok, adatbázisok mozgathatóságát és egy jól definiált gépfüggetlen felületet biztosít maguknak a felhasználóknak.

Tekintsük most át, hogy az IBM hogyan kíván ezekhez a feltételekhez alkalmazkodni.

1. *Ha egy széleskörben elterjedt szabvány létezik, akkor azt támogatjuk.*
2. *Amíg egy szabvány kialakítása folyamatban van, mi abban részt veszünk és maximális figyelemmel vagyunk felhasználóink már meglévő rendszereire. (Customers investment protection)*
3. *Ahol nincs bevezetett szabvány, ott az IBM technológiája értéket ad a felhasználónak, és ezzel együtt biztosít ehhez a technológiához más rendszerekből hozzáférési lehetőséget.*

Mennyiben tekinthető tehát nyílt rendszernek az IBM AIX operációs rendszere?

Az AIX operációs rendszer elsődleges céljai között szerepelt, hogy biztosítsa a RISC/6000-es számítógépek üzemeltetését olyan környezetben, ahol más gyártók gépeivel kell kommunikálni. Így része a rendszernek a TCP/IP, a Network File System és a

Network Computing System. Ezen lehetőségek Ethernet vagy Token Ring lokális hálózaton keresztül hozzáférhetők. Távoli helyekkel történő kommunikációra aszinkron vonalak és X.25 vonalak használhatók fel.

Az AIX ezen felül kiterjedt támogatást nyújt más IBM operációs rendszerekhez. Elsősorban az SNA nagygépes hálózati protokoll támogatását, illetve a 3270-es terminálemulátort érdemes megemlítenünk. Támogatott a RISC/6000-es számítógép PC-s Novell hálózatba illesztése is, ekkor Novell hálózati szerverként UNIX file-funkciókat és nyomtatási lehetőségeket kínál a gép.

Az AIX alapja a UNIX System V rendszer, emellett azonban követi A POSIX ajánlásait is. Megtalálhatók benne a BSD UNIX 4.3-as verziójához tartozó rendszerhivások is.

A rendszer C fordítója alapvetően a Kernighan-Ritchie definíciót követi, ugyanakkor természetesen teljesíti az ANSI C által megfogalmazott kívánalmakat. Opcionálisan vásárolható FORTRAN, Cobol, Pascal, C++ és Ada fordítóprogram is.

A grafikus alkalmazások átvitelét megkönnyítendő, a háromdimenziós grafikus fejlesztői környezet tartalmazza a GKS-t, az IBM által támogatott graPHIGS grafikus szabványt, illetve a Silicon Graphics Inc. Graphics Library (GL) nevű rutincsomagját.

Az AIXWINDOWS grafikus felület az X Window IBM-es megvalósítása. Az AIXWINDOWS-ra épülő Motif környezet az OSF ajánlásain túl támogatja a DisplayPostscript formátumot, illetve A Silicon Graphics GL alkalmazásait. Az AIXWINDOWS ezen felül ellátja a felhasználót egy desktop manager-rel, ami szemléletesebbé teszi és meggyorsítja az egyes file-okon értelmezett műveletek elvégzését.

Nézzük most azt meg, hogy a RISC/6000-es számítógép milyen hardver-tulajdonságokkal bír, és hogy a számítógép más gépekkel - elsősorban IBM termékekkel - összehasonlítva, mikor ajánlható a felhasználónak.

A RISC/6000-es gépcsalád

A RISC technológia az IBM fejlesztéseinek az eredménye. A RISC/6000-es gépcsaládban megtalálható POWER processzorokat második generációs RISC processzoroknak nevezi a szakirodalom. Ez azt jelenti, hogy a POWER (Performance Optimization With Enhanced RISC) egységek jellemzői közé nemcsak a csökkentett utasításkészlet tartozik, hanem az utasítások végrehajtási idejének gépi ciklusokra vett optimalizálása is. Ennek eredményeképp a POWER architektúrájú központi egységek öt utasítás végrehajtására képesek egy gépi ciklus alatt. Ebből a lebegőpontos aritmetikai egység egy ciklus alatt egy lebegőpontos szorzás és egy összeadás elvégézésére képes, míg a fixpontos aritmetikai egység egy fixpontos szorzást tud elvégezni. Ennek a felépítésnek az eredménye az elismerten legjobb lebegőpontos aritmetikai teljesítmény, ha a POWER architektúrát más típusokkal hasonlítjuk össze. A három RISC/6000-es családhoz (desktop, deskside és rack-mounted) különböző processzorok tartoznak, de ezek mindegyike binárisan kompatibilis a másikkal. A ciklusidőre vett optimalizálás eredménye az is, hogy a processzorok alacsony órajel-frekvenciák mellett nyújtják a fenti teljesítményt. Például a 970-es, legnagyobb RISC/6000-es alapú számítógép POWER processzora 50 MHz-es órajelet használ, és ugyanakkor a világon elsőként haladta meg ez a gép egy processzoros architektúraként a 100 SPECmark-os teljesítményt (100.3 SPECmark).

A RISC/6000-es különböző modelljeit két nagyobb csoportba szokták sorolni attól függően, hogy milyen a javasolt felhasználási terület. A **POWERstation** gépek elsősorban egy felhasználó számára nyújtanak egy multitaszkos és rendszerint grafikus környezetet. Természetesen a hálózaton keresztül történő más erőforrások igénybevétele itt is lehetséges. A **POWERserver** gépeket úgy képzelhetjük el, mint ami egy elosztott erőforrás más gépek számára. Ez a számítógép szolgálhat tradícionális sokfelhasználós szerverként, hálózati szerverként vagy egyidejűleg mindkét funkciót is elláthatja. Soroljunk fel néhány hálózati szerver-funkciót:

-File-szerver

-Számítási feladatokhoz egyfajta aritmetikai szerver

-Print-szerver

-Kommunikációs szerver (gateway)

-Help és dokumentációs feladatok szervere.

Az IBM RISC/6000-es családjában a kisebb és a közepes gépek a POWERstation és POWERserver jellegű feladatok mindegyikének ellátására készültek. Ezen belül a 200-as, illetve 300-as sorozat tagjait (ezek a desktop modellek) elsősorban egyfelhasználós, multitaszkos környezetban használják. Az 500-as gépek és a nagyobb modellek (Model 950 és 970) leginkább hálózati szerver funkciókat látnak el. Ennek megfelelően a nagyobb sorozatszámú számítógépek lényegesen nagyobb alapkiépítéssel érkeznek, illetve a bővítési lehetőségek is jóval nagyobbak (több memória, nagyobb fixlemzes kapacitás, több SCSI-eszköz csatlakozási lehetőség, stb).

Szólnunk kell a RISC/6000-es gépek grafikus lehetőségeiről is. A nyílt rendszerek kialakulását elsősorban a grafikus munkaállomások kényszerítették ki, és az AIX operációs rendszerhez tartozó, korábban már említett alkalmazások, bagy például az AIXWINDOWS grafikus felület igénylik a jó minőségű grafikus alrendszerek meglétét. Most felsorolásszerűen ismertetjük az IBM által kínált grafikus hardverek főbb jellemzőit. Mindegyik grafikus adapter közös tulajdonsága az 1280X1024-es felbontás.

A *Graycsale Graphics Display Adapter* egyidejűleg 16 szürkeségi árnyalat megjelenítésére képes.

A *Color Graphics Display Adapter* 256 színárnyalatot kezel. Az aktuális színek a true-color (8,8,8) RGB színekből állíthatók. Hardver szinten támogatja a képernyőn vonalak húzását.

A 220-as, legkisebb modellhez tartozik a *Gt1*-es jelzésű grafikus adpter. Alapkiépítésében kétszínű kártya, amit további memóriákkal négy, illetve nyolc bitsíkos, színes adapterré bővíthetünk. Alkalmas desktop publishing és CASE alkalmazások esetén.

Még a kisebb teljesítményű grafikus hardverek közé sorolhatjuk a *GT3*-as típusjelű kártyát. Ez is 16 millióból választhatóan egyidejűleg 256 szín megjelenítésére képes. A hardver támogatja a legfontosabb kétdimenziós feladatok (2D vonalhúzás, téglalap kitöltése, raszteres szöveg megjelenítése, stb) gyors elvégzését. A nyolc alapsíkon túl két overlay síkot is tartalmaz, ezen felül pedig két colormap-pel és a "bit block transfer" lehetőséggel is ellátták. Ebből a kártyából egyidejűleg kettő is elhelyezhető a 220-asnál nagyobb modellekben.

Háromdimenziós alkalmazásokra készült a *High-Performance 3D Color* adapter, ami már háromdimenziós funkciókat is megvalósít hardver szinten. Elsősorban a háromdimenziós vonalhúzást és a Goroud-árnyalt, kitöltött háromszög-generálást, a dithering és az antialiasing funkciókat fontos megemlíteni.

A Gt4 és Gt4x grafikus alrendszerek további bővítést jelentenek az előzőekhez képest. Ez a bővítés a nagy sebességet igénylő alkalmazásoknál (tervezés-automatizálás, animáció, molekula-modellezés, számításigényes képgenerálás, stb) célszerű. A kártyák egyaránt támogatják a két- és háromdimenziós funkciókat és jóval nagyobb sebességgel végzik el azokat, mint a High Performance 3D Color adapter.

Utolsónak maradt a *GT0* adapter rövid bemutatása. Ez a berendezés nem egy belső csatlakozót foglal, el, hanem egy különálló dobozként illeszthető a RISC/6000-es számítógéphez. Igen gyors számítógépes grafikai alkalmazásoknál ajánlott, különös tekintettel a számítógépes animációra. A generált képek videón való rögzítése minden grafikus adapternél támogatott.

Mivel a RISC/6000-es a grafikus munkaállomások között vált közkedveltté és a grafikus alkalmazások rohamosan változtatják meg az ember-gép kapcsolatot, az egyes programok felhasználói felületét és esetenként az egész gyártási technológát teszik szemléletesebbé, így tekintsük át az imént felsorolt grafikus kiegészítő hardverek közül a kedveltebbek néhány további paraméterét, ezúttal táblázatos formában:

	Color	HP 3D Color	Gt3	Gt4	Gt4x	GT0
Alap/Max. bitsíkszám	8/8	8/24	8/8	8/24	8/24	8/24
Alap/Max. színek száma	256/256	256/16M	256/256	256/16M	256/16M	256/16M
Double buffer síkok	0	12	0	8/24	8/24	8/24
Z-buffer síkok	-	24	-	24	24	24
Fényforrások száma	0	8	0	8	8	8
Depth Cueing	No	Yes	No	Yes	Yes	Yes
Antialiasing	No	Yes	No	Yes	Yes	Yes
2D vektorok/sec	125K	90K	650K	650K	800K	990K
3D vektorok/sec	-	90K	-	400K	800K	990K
Goroud-háromszögek/sec	-	10K	-	20K	80K	120K

A következőkben összehasonlítjuk a RISC/6000-es gépeket más IBM termékekkel, és megvizsgáljuk azt, hogy mikor melyik számítógép használata célszerűbb.

Géptípusok összehasonlítása

RISC/6000-IBM PS/2

Vessük össze elsőként a RISC/6000-es gépeket az IBM által ajánlott nagyobb teljesítményű személyi számítógépcsaládddal, az IBM PS/2-es sorozattal. A legnagyobb PS/2-esek teljesítményben átlapolják a legkisebb RISC-gépeket. Általánosságban elmondható, hogy kisebb CAD alkalmazásokhoz, irodai feladatok megoldásához, meglévő DOS-alkalmazások futtatására javasolt a PS/2. A felhasználónak megvan a lehetősége a DOS, az OS/2 és a Microsoft Windows operációs rendszer közötti választásra - a számítógépén egyidejűleg több rendszer is installálva lehet. Nem köztudott, de mindenképpen fontos megemlítenünk, hogy az IBM ajánlja az AIX PS/2 operációs rendszert is a PS/2-es számítógépeihez.

Nagyobb feladatok esetén a RISC/6000-esek teljesítménye lényeges előnyöket biztosít a felhasználó számára. Ha egy környezetben várható, hogy az erőforrások iránti igény gyorsan növekszik, úgy célszerűbb a munkaállomást választani abban az esetben, ha a feladatot csak egy nagyobb PS/2-es tudná egyébként elvégezni.

Az OS/2-es számos kiegészítéssel bír, melyekkel jól integrálható egy nyílt rendszeres architektúrába:

Network File system - adatmegosztási lehetőség más hálózati gépekkel.

X Window System - szerverként működés egy RISC/6000-esen futó kliens számára.

Remote printer support and procedure call - Távoli erőforrások (printer, másik gép számítási teljesítményének) kihasználása.

RISC/6000-AS/400

A RISC/6000-es és az AS/400-as számítógépek jelentik a "midrange" kategóriát az IBM-nél. Az AS/400-as elsősorban üzleti, irodai alkalmazásokra készült, és az elmúlt évek legsikeresebb középkategóriájú gépévé vált.

Összehasonlítva a két géptípust, a RISC-gépek javára szólnak az alábbi tulajdonságok:

Támogatás UNIX-alkalmazásokhoz, melyek száma mára igen jelentős lett.

Numerikus számításigényes feladatok gyors elvégzésének lehetősége.

Grafikus képességek.

Szabványos felületek más gyártók gépeinek csatlakoztatására.

Az AS/400-as számítógépek előnyei az alábbi területeken mutatkoznak meg:

Integrált relációs adatbáziskezelő és lekérdező funkciók az operációs rendszerben.

Nagy online tárolási kapacitás, ami elosztott rendszerbe is illeszthető.

Tranzakció-processzálási lehetőség.

SNA és APPN kapcsolati lehetőség (Az APPN az SNA módosítása egy nem hierarchikus architektúra irányába).

Könnyű használat.

Azon felhasználók számára, akik AS/400-as gépüket más rendszerekbe kívánják integrálni, a következő lehetőségek adottak, illetve állnak fejlesztés alatt:

X-kliens alkalmazás támogatása - ezáltal bekapcsolódni egy elosztott X Window környezetbe.

Hálózati kommunikáció TCP/IP felületen, beleértve a Network File System lehetőségét is.

OSF által definált "remote procedure call" és POSIX IEEE 1003.1 és 1003.2 ajánlásainak megvalósítása.

RISC/6000-ES architektúrák

Az IBM nagygépei teljesítményben, felépítésben teljesen különböznek a középkategóriájú munkaállomásoktól, így egy ilyen összehasonlítás értelmetlen. Itt szeretnénk azonban röviden ismertetni az **AIX/ESA** rendszert, ami az IBM nagygépes UNIX megoldása.

Általánosságban a mainframe-ek előnyei között az alábbiakat említhetjük meg:

Nagyobb leterhelhetőség a felhasználók, illetve az alkalmazások számát illetően.

Nagysebességű I/O-műveletek.

Nagykapacitású, alacsony költségű adattárolás.

24-órás rendelkezésre állás.

Számításigényes és/vagy üzleti programok egyidejű futtatása.

Többszörös adatvédelmi mechanizmusok.

Azon felhasználó számára, akik a nagygépes előnyöket szeretnék összekapcsolni egy UNIX-os környezettel, az IBM 1992 júniusa óta ajánlja az AIX/ESA operációs rendszert. Az AIX/ESA-val az IBM az alábbi feladatokat szeretné ellátni:

-Numerically Intensive Computing

-Data Server

-Campus Server

A UNIX-rendszer nagygépes lehetőségeit különösen egyetemi környezetben érdemes hangsúlyozni. Lehetővé válik igen nagyméretű programok futtatása, amihez a megbízhatóságot az ES/9000-es architektúra biztosítja. Az operációs rendszer képes a

többprocessoros számítógépen az erőforrások optimális kihasználására, beleértve a vektorprocesszoros bővítéseket is. A külvilág felé a nyílt rendszereken megtalálható adatkapcsolati, hálózati protokollok segítségével történhet a kommunikáció. Az X Window implementálásával lehetőség van a grafikus, interaktív környezet lehetőségeinek kihasználására, grafikus alkalmazások futtatására. Az AIX/ESA követi az OSF legújabb ajánlásait, melyek egy,a későbbiekben kialakuló egységes operációs rendszer irányába mutatnak.

Külön gondot fordít az IBM az adatbiztonság nagygépes-szintű megteremtésére . A rendszeradminisztrátori feladatok különválnak, és nagyobb az egyes felhasználói hozzáférési jogok hangolhatóságának lehetősége. Kiterjedt elszámolási, naplózási lehetőségek biztosítják a rendszer állandó lekérdezhetőségét, figyelését.

A programfejlesztési eszközök megegyeznek a megszokott UNIX-beliekkel. A számítógép a külvilág felé egy teljesen "normális" UNIX-munkaállomásnak látszik, ami a belső felhasználó számára kiegészül egy nagy teljesítmény érzékelésével.

Nagygépes rendszereknél, távoli felhasználók esetén a távoli terminál és a központi gép közötti adatátvitel lassúsága komoly késleltető lehet. A UNIX-terminálokra mindez különösen igaz, mivel ezek minden egyes karaktert elküldenek feldolgozásra. Az AIX/ESA esetében ennek kiküszöbölésére lehetőség van közbülső RISC/6000-esek beiktatására, amelyek a terminálról érkező információ jelentős részét megszűrik, mintegy saját hatáskörükben döntenek. Ez tehermentesíti a kommunikációs csatornát.

Számíthatunk arra, hogy a jövőben értékesítendő ES/9000-es gépeken belül egyre nagyobb arányban lesznek AIX/ESA operációs rendszerek installálva. Mindez a számítógépes programrendszerek előállítóit arra sarkallta, hogy termékeikkel ezen a platformon is megjelenjenek. Várható tehát, hogy rövid időn belül a legelterjedtebb ügyviteli és adatbáziskezelői programok megjelennek ezen az operációs rendszeren is.

Összefoglalás

Reméljük, hogy írásunkban sikerült újabb információkat adnunk mindazoknak, akiket érdekelnek a nyílt rendszerek és kíváncsiak az IBM ilyen irányú törekvéseire. Biztosak vagyunk abban, hogy a jövőben mindenki számára igazolást nyer: az IBM és a nyílt rendszerek egyre szorosabban összetartozó fogalmak.

Készült a PRINT PACK Nyomdában, Fv.: Báder György

HUUG - Hungarian UNIX Systems' User Gro

Titkárság: Neumann János Számítógéptudományi Társaság * Cím: 1054 Budapest V., Báthori
Levélcím: 1368 Budapest 5, Pf.: 240. * Telefon: 132-9349, 132-9390 * Fax: 131-8140 * Telex: 22-57