941484

Nemzeti Információs Infrastruktúra Fejlesztési Program

Információs Füzetek 11. 1.

DRÓTOS LÁSZLÓ · POPOVICS PÉTER

HOGYAN CSINÁLJUNK SAJÁT GOPHERT?

Budapest

1995

Drótos László

Popovics Péter

Hogyan csináljunk saját Gophert ?

128200

102508

Drótos László

Popovics Péter

Hogyan csináljunk saját Gophert ?

II. 1.

MTAK

N.I.I.F.

Budapest, 1995

941484

NIIF Információs Füzetek II. 1.

© Drótos László (h1192dro@ella.hu)

© Popovics Péter (Peter.Popovics@eik.bme.hu)

Sorozatszerkesztők:

Drótos László (h1192dro@ella.hu)

Kokas Károly (kokas@bibl.u-szeged.hu)

Lektor:

Turchányi Géza (turchanyi@sunserv.kfki.hu)

MAGYAR TUDOMÁNYOS AKADÉMIA KÖNYVTÁRA

ISBN 963 02 9795 7

Kiadja a Nemzeti Információs Infrastruktúra Fejlesztési Program Koordinációs Iroda

NIIFKI vezetője: Nagy Miklós

A kiadásban közreműködött: Kornétás Kiadó

Ügyvezető igazgató: Pusztay Sándor

Műszaki szerkesztő: Gáspár Imre

Nyomta: Komáromi Nyomda és Kiadó Kft., Komárom

95-5

Felelős vezető: Kovács Jánosné ügyvezető igazgató

M. TUD. AKADÉMIA KÖNYVTÁRA

Könyvleltár 7308/19 95 sz.

Tartalom

Gopher és gopher+ — röviden a történetről / 7
A gopher születése / 7
Gopher+ — Pocok és a zsákja / 9
A gopher rendszer működésének lényege / 10

Az információs szolgáltatás megtervezése / 13
Tartalom és forma / 13
Néhány általános vezérelv / 14

A gopher szerver program telepítése / 17
Honnan lehet beszerezni a gopher programokat? / 17
Jogosultságok és biztonsági kérdések / 17
Hogyan kell telepíteni a gopher programokat? / 19
A gopher szerver elindítása / 25

A gopher feltöltése adatokkal / 29
A szerver viselkedése különféle típusú fájlokkal / 29
Hivatkozások („link"-ek) / 30
Az eredeti nevek átírása a .cap directory segítségével / 31
Linkek létrehozása / 31
Programok futtatása / 32
A gopher adatterület egyes részeinek elrejtése / 33
Teljesszövegű indexek / 33
Ask blokkok / 35
WWW és HTML támogatás / 36

A gopher kliens — nyilvános gopher kliens szolgáltatások / 37
A kliens program szintaktikája / 37
Konfigurációs fájl / 37

Gopher magyarosan / 38
Kliens oldali dekódolás / 38
Szerver oldali konverzió / 39
Gopher a tükörben – gmirr / 39
Magyarországi konvenciók / 40

Menedzselés és propaganda / 42
Néhány jó tanács a szerkesztés koordinálásához / 42
Hírverés / 43
Gopher vagy WWW? / 44

Ajánlott információforrások / 45

1. melléklet: A Gopher+ protokollban szereplő attribútumok listája / 47

2. melléklet: A legfontosabb és leggyakrabban használt gopher-típusok / 50

Gopher és gopher+ — röviden a történetről

gopher n. – 1. Any of various short tailed, burrowing mammals of the family Geomyidae, of North America. 2. *(Amer. colloq.)* Native or inhabitant of Minnesota: the Gopher State. 3. *(Amer. colloq.)* One who runs errands, does odd–jobs, fetches or delivers documents for office staff. 4. *(computer tech.)* Software following a simple protocol for tunneling through a TCP/IP internet.

A gopher születése

A Minnesotai Egyetemen 1991 elején elkezdtek tervezni egy intézményi szintű információs rendszert. Az eredeti elképzelésekben egy nagy és összetett szoftver szerepelt, ám a Számítóközpont néhány munkatársa (érdemes a nevüket megjegyezni: *Farhad Anklesaria, Paul Lindner, Daniel Torrey, Bob Alberti* és *Mark McCahill)* előállt egy jóval egyszerűbb rendszerrel, amit maguk között „gopher"-nek neveztek. Az általuk javasolt megoldás túl egyszerűnek tűnt az egyetem vezetése számára és végül egy külső cégnek adták ki a fejlesztést. Szerencsére a gopher-csapat nem adta fel, s rövidesen létrehozott egy működő prototípust, amivel könnyen el lehetett érni a hálózaton levő információs anyagokat. Néhány hónap alatt kiderült, hogy a nagy és komplex rendszer sohasem fog elkészülni, a gopher viszont közben népszerűvé vált az egyetemen és rövidesen az egész világon, s így gyorsan kialakult a felhasználók „kritikus tömege".

A gopher – a legtöbb mai hálózati szoftverhez hasonlóan – két fő részből áll: egy, a felhasználót támogató ún. kliens programból, és egy, az adatokat szolgáltató ún szerverből. A kliens megnyit egy kapcsolatot a szerver felé, bizonyos információkat kér tőle, megkapja és – miután lebontotta a kapcsolatot – megjeleníti azokat. Az első verzióban csak kétfajta információforrás létezett: szövegfájl és „iratgyűjtő" (alkönyvtár). Vagyis a fejlesztők már az elején egy olyan mechanizmusban gondolkodtak, amely kiválasztott állományokat és directory-kat továbbít. A gopher filozófiájának ez a sajátossága máig is megmaradt.

1991 április végén a gopher fejlesztői úgy döntöttek, hogy valamiféle kereső mechanizmusra is szükség van. A NeXT munkaállomások tűntek a legalkalmasabb szervereknek ebből a szempontból, mert ott az alapszoftverbe épített Digital Librarian nevű program nagyon jól használható keresőrendszerrel rendelkezik. A programozók azt is elhatározták, hogy a gophernek támogatnia kell az elektronikus telefonkönyvekben való keresést is, és megegyeztek abban, hogy a CSO-féle protokollt fogják alkalmazni erre a célra, melyet már akkor is több egyetemen használtak, mint a legjobb módszert az online telefonkönyvek megvalósítására.

Ezek a különböző dokumentumtípusok – szöveg, directory és CSO keresés – elegendőek ahhoz, hogy egy gophert definiáljunk, amely egy adott szerverről az ott levő dokumentumokat szolgáltatni tudja. A protokoll azonban már kezdettől is megengedte, hogy egy szerver egy másik szerverre mutasson, amelyen egy bizonyos dokumentum fizikailag megtalálható. Ez lehetővé teszi a gophernek, hogy az Interneten szétszórt szolgáltatásokat és információforrásokat egyetlen listában mutassa meg. Arra is lehetőség van, hogy a többi gopherről is megjelenítsen egy listát, melyből bármelyik egyetlen gombnyomással vagy egérkattintással kiválasztható.

A fejlesztők arra is gondoltak, hogy valamilyen módon az egyetemi hálózaton már létező adatbázis-szolgáltatások is elérhetők legyenek a gopherből, mint például az egyetemi könyvtár online katalógusa. Elhatározták, hogy bevezetnek egy újabb dokumentumtípust, amely valójában egy telnet hívás egy másik számítógépbe. Hogy teljes legyen a kínálat, bevették a Macintosh és a PC bináris állományait is. Az eredeti protokollt bővíthetőnek tervezték; az egy byte-os dokumentumtípus mezővel elvileg 255-féle különböző információforrás-típust lehet leírni. A gopher protokoll első vázlatos leírása 1991 tavaszán került közzétételre.

A hang átvitelének lehetősége a gopher segítségével már a fejlesztés egészen korai fázisában megszületett. Az egyik hétvégén ugyanis Paul Lindner, aki a szerver programjának nagy részét írta, egy kis zenét szeretett volna hallgatni az irodájában, mely több szobányira van a nyilvános CD lemezjátszótól. A NeXT gépe tudott hangot megszólaltatni és ott volt egy CD olvasó egy másik munkaállomásra kötve a másik szobában. Néhány órával később bevezette a „hang" állománytípust és attól kezdve a gopher képes volt zenét szolgáltatni valós idejű Internet kapcsolaton keresztül.

Az első gopher szolgáltatások 1991 nyarának végén indultak meg a Minnesotai Egyetemen. A világnak 1991 júliusában jelentették be a gopher megszületését egy, az egyetemi információs rendszerekkel foglalkozó levelezőcsoporton keresztül (cwis-l@wuvmd.bitnet). Létrehozták az alt.gopher nevű Usenet newsgroupot is. A számítógépes rendszergazdák szerte az Interneten elkezdtek megismerkedni a gopherrel. A forrásprogramot mindenki számára elérhetővé tették kipróbálásra, és a legkülönbözőbb helyeken kezdtek felbukkanni a gopher szerverek. 1992 elejére a gopher többé már nem egy kísérleti prototípus volt, hanem egyre inkább egy választható alternatíva lett azoknak az egyetemeknek a számára, amelyek szerettek volna egy helyi információs rendszert létrehozni.

A gopherről az első magyar nyelvű ismertés az NJSzT 1992-es nyári konferenciáján hangzott el, Tim Berners-Lee (a nagy konkurrens, a World Wide Web atyja) és Turchányi Géza előadásában: „Hogy legyen olcsó, ami ingyen van". Az első gopheres hírek és cikkek itthon a KATALIST@HUEARN.BITNET levelezőcsoportban bukkantak fel. Kísérleti gopher szolgáltatás hazánkban először a KFKI Internet Klubjának kezdeményezésére indult – szintén romantikus körülmények között –, 1992 őszén. Ezt követte az X.25 hálózatról is elérhető nyilvános IIF gopher kliens a mars.sztaki.hu gépen, majd a BME gopher szerver indítása. (– a lektor kiegészítése)

Gopher+ — Pocok és a zsákja

Idővel az egyszerű rendszerek is továbbfejlődnek, csak az a kérdés, hogy közben meg tudnak-e maradni egyszerűeknek és átláthatóaknak. Egy 1992 augusztusában tartott konferencián mutatták be a minnesotai fejlesztők a **gopher+** változatot, az eredeti terv továbbfejlesztésére vonatkozó új elképzeléseiket. Időközben ugyanis nagyon sokféle igény és javaslat merült fel az eredeti protokoll bővítésére, hogy kezelni lehessen mindent a PostScript fájloktól kezdve a különböző képformátumokig (pl. GIF vagy JPEG), és hogy egyéb attribútumokat is lehessen rendelni a dokumentumokhoz, mint például a szerző neve és a dokumentum érvényességnek határideje. Ahelyett, hogy mindezeket a javasolt kiterjesztéseket a protokollba építették volna bele, a Minnesotai Egyetem gopher fejlesztői egy olyan mechanizmust találtak ki, amivel általánosan megoldható az új elemek bevezetése a gopher szabványba.

A gopher+ már új, névvel ellátott mezőkkel egészíti ki az eredeti gopher egykarakteres típuskódját. Ha a kliens egy felkiáltójelet tesz a selector string mellé, akkor a gopher szervernek egy ún. Attribute Information blokkot is el kell küldenie a válasszal. Az első névvel rendelkező blokk, a +INFO feltétlenül szükséges; ez hasonlít a régi protokoll által használt információs adatsorhoz. A gopher+ esetében további mezőket is javasoltak, mint például az +ABSTRACT (a dokumentum összefoglalója), az +ADMIN (a dokumentumért felelős személy adatai), a +DATE (a fájl utolsó módosításának időpontja), és más hasonlók. Ezek az általános attribútumok a rendszerfelügyelőket segítik a dokumentumok karbantartásában és adminisztrálásában, és – a kliens programok megfelelő továbbfejlesztése esetén – a felhasználóknak is nagy segítséget jelentenek majd az őket érdeklő információs anyagok kiválasztásában.

A University of Minnesota fejlesztői felvállalták a gopher+ állománytípusok központi nyilvántartását; bárki, aki egy új nevű attribútum mezőt akar bevezetni, a gopher teamnél regisztráltathatja azt. Ezzel a módszerrel együttműködve lehet új mezőket bevezetni, amennyiben a gopher közösség belegyezik. Elvileg arra is lehetőség van, hogy egy adott helyen olyan mezőket is bevezessenek a helyi kliensekhez és szerverekhez, amelyek mások számára nem érdekesek. Mint eddig is, a kliens csak azokat az információkat fogadja el, amelyekről tudja, hogy hogyan kell kezelni őket. Ha valaki létrehoz egy +HAIRCOLOR attribútumot, akkor a gopher+ kliens ezt nyugodtan figyelmen kívül hagyhatja. Egy gopher+ mező lehet egy egyszerű szöveges név, vagy – akárcsak a dokumentumok esetében az eredeti protokollban – egy másik gopher+ szerverre való mutató is.

Egy másik fontos újdonság, hogy a gopher+ képes egy dokumentum különböző formátumú változatait a felhasználó kívánsága szerint megjeleníteni. A +VIEWS nevű mező segítségével választani lehet például egy szöveg több nyelvű vagy különböző formátumú (pl. sima ASCII vagy PostScript) változatai közül.

Az attribútumok és a választható megjelenítési formátumok mellett a gopher+ már interaktív módon kitölthető űrlapokat is tud kezelni. A gopher+ szerver meg tud kérdezni a felhasználótól különböző adatokat (pl. jelszót), sőt közvetítőként is működhet a felhasználó és egy másik adatbázis között. Az ún. +ASK blokkok segítségével meg lehet kérdezni a felhasználótól mondjuk egy fájl nevét, vagy felszólítani, hogy válasszon a felkínált opciók közül.

A gopher+ protokollt kihasználni képes első kliensek 1993 februárjában jelentek meg: a Mac TurboGopher kliensének egy újabb verziója és egy új Unix kliens. Azóta természetesen már számos gopher+ alkalmazást fejlesztettek különböző platformokra (pl.: a Minuet kliens DOS alá, vagy a HGopher a Windowshoz). A nem gopher+ klienst használók is nyugodtan választhatnak gopher+ szervereket, de nem tudják a gopher+ dokumentumokat megjeleníteni.

Közzétettek egy RFC dokumentumot az „alap" gopher protokoll leírásával (RFC 1436). Mark McCahill szerint ezen a dokumentumon már csak kisebb változtatások lesznek, majd ezután az eredeti protokollt „befagyasztják". A gopher+ leírása pedig megtalálható a University of Minnesota gopher szerverén.

A gopher rendszer működésének lényege

A gopher szerverek szenzációs egyszerűséggel működnek: lényegében a szolgáltató gép fájlrendszerét vetítik ki a felhasználó felé a gopher protokoll szerint. A felhasználói oldalon futó program (a kliens) jegyzi meg azt, hogy mely szolgáltatóktól milyen információforrások kérhetők és szükség esetén újra kapcsolatba lép a kívánt szolgáltatóval, s elkéri a megfelelő információt. Eközben a szerver semmilyen adatot nem tárol arról, hogy ki és mit kérdezett tőle utoljára (legfeljebb egy statisztikai célokat szolgáló „log" állományban), azaz megmarad „állapot függetlennek" (angolul: *stateless*). A gopher elvének ez a sajátossága magyarázza a gopher nagy hatékonyságát – a szerver csak annyi ideig van a felhasználóhoz kötve, amennyi egy adott kérés teljesítéséhez kell, és nem kell megküzdenie azzal a terheléssel, amit az okozna, ha több száz vagy esetleg több ezer felhasználó lenne bejelentkezve egyszerre. Egy ilyen egyszerű rendszert könnyű javítani is: ha egy szolgáltatóprogram elszáll, elég újra indítani és minden mehet tovább.

A gopher információs bázisa egy könyvtárban foglal helyet. Ha a szolgáltató gépen elindítjuk a szerver programot, közölni kell vele ennek a könyvtárnak a nevét. A szerver program elkezd figyelni egy adott TCP/IP portra. Ezek után, ha egy kliens program csatlakozik és a főmenüt kéri, a szerver elküldi a kijelölt könyvtár tartalmát, a gopher protokoll szerint. Ami a szerver oldalán alkönyvtár volt, az a kliens oldalon almenüként jelenik meg, ami a szerver oldalon egy fájl volt, az a kliens oldalon letölthető állományként tűnik fel a menüben. Az eredeti gopher protokoll nagyon egyszerű, mindenkinek javaslom, hogy próbálja ki a következő kis játékot, hogy megértse a működését:

- Telnetteljen be egy olyan gépre, amin fut gopher szerver, a 70-es portra. Például:

```
Telnet gopher.bme.hu 70
```

- Miután felépült a kapcsolat, nyomja meg az *Enter* gombot. Tulajdonképpen ezt „csinálja" a gopher kliens is, amikor először csatlakozik egy gopher szolgáltatóhoz; egy üres *selector*-t küld el. Erre a szerver a főmenü elemeinek leírásával válaszol:

```
1English information        1/engl goliat.eik.bme.hu   70 +
1Magyar nyelvű információk  1/hun  goliat.eik.bme.hu   70 +
1- Karakterkészlet választás--     goliat.eik.bme.hu   70
1ISO Latin-2                       goliat.eik.bme.hu   70
1PC 852                            goliat.eik.bme.hu   7001
1Ekezet nelkuli                    goliat.eik.bme.hu   7002
1CWI                               goliat.eik.bme.hu   7003
1Repulo ekezetes                   goliat.eik.bme.hu   7004
```

Ezeknek a soroknak a felépítése a következő:

```
<Type><Document Title>\t<Selector>\t<Host>\t<Port>
```

Minden egyes dokumentumhoz egy ilyen adatsor tartozik. Az első adat egy egykarakteres kód a dokumentum típusának jelzésére, amelyet a gopher protokoll definiál. A legalapvetőbb típusok:

```
0   -   Text File
1   -   Directory
9   -   Bináris fájl
s   -   Hang fájl
I   -   Kép
```

A típust jelző karakter hozzá van kapcsolva az őt követő címhez; a többi adatmező az ASCII kódtábla TAB karakterével van elválasztva (\t). A *Document Title* mező az az ismertető szöveg, amit a kliensnek az egyes tételeknél meg kell jelenítenie. A *Selector* egy olyan karaktersorozat, mely rendszerint a dokumentum helyére utal az adott szerver fájlrendszerén belül, és a dokumentum egyértelmű azonosítására szolgál, amennyiben szükségessé válna a letöltése. A *host* egyszerűen a szerver teljes címe a hálózaton, a *port* pedig az a TCP/IP port, amelyen a *host* várja a „gopher hívásokat". (A 70-es csatorna a szabványos gopher port, bár egy adott szerver más portokat is használhat, amennyiben egy gépen egyszerre több gopher szolgáltatás is fut.)

A kliens minden dokumentum esetén megvizsgálja a típust jelző egykarakteres kódot, ami alapján kiválasztja a megfelelő megjelenítő (vagy például megszólaltató) rendszert. Ha a kliens nem képes megjeleníteni az adott tételt (pl. egy VT terminál nem tud grafikát és hangot kezelni), nem tünteti fel a menüben, vagy ha

mégis, csak a kimentését teszi lehetővé, „megmutatását” nem. Ez bizonyos esetben hasznos lehet, például ha a felhasználó szeretné letölteni a fájlt a kliensen keresztül és később, már a gopheren kívül használni azt.

A gopher egyik legnagyobb újítása, hogy a szerver program más szolgáltatókról is tud információkat küldeni a felhasználónak. Vagyis ahelyett, hogy a szolgáltatók láncban adogatnák egymásnak az adatokat, egyszerűen csak az közlik a felhasználóval, hogy hova kapcsolódjon a kliensével, ha egy bizonyos információt egy másik szolgáltatótól akar lekérni. Így a szerverek egymásra mutogatnak és a felhasználói programok a mutatók („link”-ek) birtokában követik az információforrásokat.

„A gopher nagyszerűen egyszerű, ettől lett egyszerűen nagyszerű.”

Az információs szolgáltatás megtervezése

Tartalom és forma

A gopher alapfilozófiája szerint hierarchikusan szervezett menük segítségével vezeti a felhasználót és mutatja az információforrásokat. Ez egyrészt azt jelenti, hogy a felhasználó mindig ugyanolyan elv szerint navigálhat a lehetőségek között (még akkor is, ha a gopher egy ftp archívumot, egy levélgyűjteményt, vagy egy keresés eredményét mutatja éppen), másrészt viszont a menürendszer bizonyos korlátokat jelent és körülményesebbé teszi a gopher használatát (pl. a WWW rugalmas hipermédia képességeihez viszonyítva). Amikor megtervezzük a gopherünk szerkezetét, akkor törekedni kell arra, hogy ezeket a kényelmetlenségeket a felhasználók minél kevésbé érezzék (pl. áttekinthető menük, keresési lehetőség a menüpontok címeiben, több szintet átugró linkek, a monoton menüket színesítő grafikai elemek, stb.). Másrészt arra is gondolni kell, hogy egy komolyabb gopherben akár több tízezer állomány és menüpont is lehet, s ez már olyan nagyságrend, aminél feltétlenül átgondolt információszervezésre, valamilyen – következetesen végigvitt – koncepcióra van szükség. Azért is fontos erre idejében gondolni, mert egy már kiépített hierarchia átstruktúrálása nagyon kellemetlen és időrabló feladat. A gopherek meglepően nagy százaléka nem éri meg az első életévét sem, mert vagy elmegy a gazdája kedve a továbbfejlesztéstől, vagy kiderül a használt struktúra alkalmatlansága vagy hiányossága, s inkább kidobják az egészet, hogy új elvek alapján egy másikat csináljanak.

A világ gopherein található információs anyagok száma már tízmilliós nagyságrendű és mind nehezebb „újat mondani". Ezért a „tartalom" mellett/helyett egyre inkább a „forma" (vagyis a könnyű használat) a fontos a felhasználók számára, ez alapján választják ki a kedvenc gopherüket a több ezer közül. Rengeteg tippet kaphatunk az információk szervezésével kapcsolatban a gopherrel foglalkozó levelező listákon és hírcsoportokban a többi fejlesztőtől; továbbá érdemes ötleteket „lopni" a legnépszerűbb, példamutatóan szervezett gopherekből; és figyelemmel kísérni a gopherünk használatát mutató statisztikákat, kikérni a felhasználók véleményét a szolgáltatásunkkal kapcsolatban (pl. „online panaszkönyv"). Végül: nem szégyen segítséget kérni olyan valakitől, aki szakmája szerint az információk rendszerezésével és szolgáltatásával foglalkozik (pl. egy könyvtárostól).

Az első eldöntendő kérdés, hogy milyen jellegű gophert szeretnénk csinálni? A jellemző típusok jelenleg ezek:

- **CWIS** (egyetemi/intézményi belső faliújság, információs rendszer)
- **közösségi** (egy körzet lakóit kiszolgáló rendszer, hálózati „kapu")
- **szakmai** (egy adott szakterület információforrásainak gyűjteménye)
- **kiadói/könyvtári** (elektronikus publikációk terjesztésére szolgál)

- **meta/index** (rendezett és/vagy kereshető átjárók más rendszerekhez)
- **hobbi/játék** (szórakoztató anyagok gyűjteménye és (online) játékok)
- **vegyes** (az előzőek tetszőleges keveréke, ez a leggyakoribb)

Az egyes típusokhoz saját optimális szerkezet (és tartalom) tartozik, ami miatt nem érdemes egy szerveren keverni őket, és amit érdemes tiszteletben tartani, megkönnyítve ezzel a felhasználók dolgát, akik így a hasonló célú rendszerekben hasonló elveken felépülő menüket találnak. Az itt rendelkezésre álló hely kevés ahhoz, hogy az egyes típusok sajátosságait részletezzük, a legjobb tanács az adott stílusban legnépszerűbb (külföldi) szolgáltatások tanulmányozása és ésszerű másolása lehet.

Néhány általános vezérelv

- A főmenü felépítésére vonatkozóan két irányzat vetélkedik: az egyik szerint a kezdő menülista minél rövidebb (max. egy képernyő) legyen, mert ez gyorsan megjelenik, áttekinthető és a felhasználót nem hozza zavarba a sokféle lehetőség; a másik a hosszabb főmenüt preferálja, mert így a keresett téma durva kiválasztása gyorsan lezajlik, az alsóbb szinteken nem kell olyan „mély", soklépcsős menüket kialakítani és a gyakran keresett szolgáltatások közvetlenül elérhetők a főmenüből. A kétféle megoldás közötti választást a saját ízlésünk mellett a szolgáltatás jellege és a felhasználó kör igénye befolyásolhatja. Általában a *közösségi* és a *meta/index* jellegű gophereknél az első típus, míg a *CWIS* és *szakmai* szolgáltatásoknál a második ajánlható. (A hazai gophereknél a főmenü gyakran a használni kívánt ékezetes kódkészlet kiválasztására szolgál. Ilyenkor a probléma értelemszerűen a második menüszintre tevődik át.)
- A főmenüben mindenképpen érdemes elhelyezni egy rövid impresszumot (a szolgáltatás bemutatása, technikai jellemzői, a szerkesztők adatai és a kapcsolatfelvételhez használható e-mail cím), valamint egy kezelési utasítást a kezdő felhasználóknak és egy keresési lehetőséget a menüpontok címeiben (ha nincs ilyen funkció, akkor legalább tegyünk ide egy, a fontosabb menük hierarchiáját mutató „térképet", vagy egy hosszabb szöveges ismertetőt a gopher tartalmáról).
- A folyamatosan aktualizált gophereknél ajánlatos egy lista vagy menü az elmúlt napokban vagy hetekben feltett új anyagokról, hogy a felhasználónak ne kelljen végigbogarásznia a menüket az újdonságokat keresve. (Természetesen ebből a listából mindig törölni kell azokat a tételeket, amiket a rendszeres felhasználók feltehetően már megnéztek és amelyek ettől kezdve már csak a „rendes" helyükön érhetők el, hogy a lista áttekinthető méretű maradjon.) Ugyancsak gyakori megoldás, hogy a gopher gazdája a főmenüben elhelyezett levélben „üzen" az olvasóknak és hívja fel a figyelmet az újdonságokra.
- A felhasználók által leggyakrabban keresett anyagokat és funkciókat a legfelső szinteken és az egyes menük elején kell elhelyezni, hogy minél kevesebb gombnyomással elérhetők legyenek. Egy másik lehetőség, hogy nem emeljük ki

őket arról a helyről, ahová logikusan tartoznak, de közvetlen, több menüszintet átugró linkeket készítünk hozzájuk valahol a hierarchia tetején (pl. „A legkeresettebbek" címmel), hogy így is elérhetők legyenek.

- Nem szabad elfelejteni, hogy a gopher egy nyilvános szolgáltatás, amit potenciálisan több millió ember tud elérni a világ minden részéről, ezért felépítésének – talán a *hobbi* típusú rendszereket kivéve – nem a készítőjének észjárását, hanem valami, mindenki számára könnyen felismerhető logikát kell követnie. Ennek a logikának lehetőleg több menün keresztül is érvényesülnie kell (pl. az első fájl mindig egy leírás a listában szereplő többiről). Kerüljük a tartalmilag nem összetartozó anyagok egy menüben való szerepeltetését (pl. egy „Érdekességek" vagy „Egyéb" című menüpont alatt), illetve az eltérő típusú vagy nyelvű fájlok keverését. Nem szerencsés továbbá az ékezetes kódkészlet közötti választásra szolgáló menüpontokat belekeverni a tulajdonképpeni főmenübe.
- Az információforrásokat lehetőleg témák szerint és ne az elérésükhöz szükséges hálózati eszközök szerint csoportosítsuk (egyre kevésbé érdemes pl. „Gopherek", „WWW-k", „FTP-k" nevű menüket csinálni, mert a mai Internet kliensek már szinte mindegyiket meg tudják jeleníteni úgy, hogy a felhasználó észre sem veszi, hogy milyen típusú szervert használ éppen.)
- A menük áttekinthetőségét növelhetjük és egyhangúságukat csökkenthetjük, ha üres sorokat vagy elválasztó vonalakat és egyéb grafikai elemeket helyezünk el bennük. De ne felejtsük el, hogy a gopherünket a legkülönbözőbb géptípusokkal, kliensprogramokkal és betűkészletekkel fogják olvasni, így egy gondosan megtervezett vizuális hatás teljes zagyvasággá válhat egy másfajta számítógépes környezetben.
- Ha a szolgáltatásunkat nem egy szűk és ismert körnek szánjuk, akkor általában törekedni kell a „legnagyobb közös osztó" elv betartására. Vagyis ne tegyünk fel túlzottan „egzotikus" formátumú állományokat vagy legalább mellékeljünk egy elterjedtebb (pl. ékezet nélküli) verziót, hogy a legegyszerűbb eszközökkel kommunikáló felhasználó is használni tudja a rendszerünket. A különösen nagy méretű állományokból készítsünk egy feldarabolt vagy összetömörített változatot is, ami könnyebben letölthető. **A gopher egyik legfontosabb előnye a WWW-hez képest éppen az egyszerűsége!**
- A másik fő különbség, hogy a gopher nem hipertext. Bár megvan a lehetősége annak, hogy különböző linkekkel össze-vissza kötögessük a gopher-menüket, mégis álljunk ellent a kísértésnek és ne csináljunk tucatnyi hivatkozást a kedvenc állományainkra, mert a felhasználó pillanatok alatt egy átláthatatlan labirintusban találja magát és elveszti azt az érzését, hogy egy egyirányú hierarchiában halad lefelé, vagy lépeget vissza. Ha olyan információgyűjteményünk van, ami sok kereszthivatkozást igényel, akkor csináljunk inkább WWW szervert.
- A gopher és WWW szolgáltatásoknál a legnagyobb frusztrációt az okozza a felhasználóknak, hogy egy-egy menüpont vagy link mögött nincs semmi, a

kiválasztott funkció a „levegőbe" mutat (néha akár minden második-harmadik állomány is hiányozhat), vagy éppen az ott található információ már régen elavult. A gopher gazdájának rendszeresen monitoroznia kell a rendszerét; ellenőrizni, hogy a linkek működnek-e még, ill. törölni az érvénytelenné vált dolgokat. A még nem létező, de a menüben már feltüntetett szolgáltatásoknál legalább egy „Fejlesztés alatt" típusú szöveget el kell helyezni az olvasók tájékoztatására. A fontosabb törölt anyagok helyét még egy jó darabig érdemes megőrizni és az állományt egy „Már nem elérhető" stílusú felirattal helyettesíteni, hiszen nem tudható, hogy hány távoli link és Veronica adatbázis mutat még rá. A csak áthelyezett menüpontoknál egy, az új helyre utaló feliratot vagy linket illik elhelyezni a régi címen (az előbbihez hasonló okokból).

- Ha a gopherrel kezelt anyagok a helyi hálózat több gépén vannak elosztva (tipikus eset *CWIS* rendszereknél), akkor a legfontosabb dokumentumokat érdemes megduplázva a központi szolgáltató gépen is elhelyezni, amiről feltételezhető, hogy mindig üzemel. (Természetesen gondoskodni kell a teljes gopher gyűjtemény rendszeres mentéséről is.) Ugyancsak érdemes alternatív lehetőségeket megadni távoli szolgáltatásokra mutató linkeknél (különösen a *meta/index* típusú gopherekben), hogy ha a távoli gép éppen nem üzemel, a felhasználó választhasson egy másik, hasonló szolgáltatást.
- A gophereket indexelő és kereshetővé tevő megoldások (pl. Veronica) többnyire csak a menüpontok nevét adják meg a keresgélő felhasználóknak. Az ilyen, környezetükből kiragadott nevek sokszor semmi támpontot nem adnak arról, hogy mi van mögöttük. Ezért a fontosabb anyagainknak és szolgáltatásainknak beszédes és minél több kulcsszót tartalmazó nevet adjunk a menükben. (Ha például azt szeretnénk, hogy az intézményünk angol nyelvű ismertetésére minél könnyebb legyen ráakadni, akkor ne egy „Introduction" vagy „Read this" menüpont mögé rejtsük el, hanem az intézmény nevét is tüntessük fel.) Az újabb szabványok már támogatják az olyan attribútumok megadását is, mint például az ABSTRACT, ahol további információ helyezhető el az illető gopher-tétel tartalmáról. Ezt szintén érdemes kihasználni.
- Saját munkánkat könnyítjük meg azzal, ha nemcsak a menük szövegét, hanem a fájl- és a directory-neveket is következetesen és értelmesen választjuk meg. Az ezernyi állomány között sokkal könnyebb lesz eligazodni, ha már az elején kidolgozunk egy rendszert a különböző fájl-típusok, nyelvek, témák stb. jelölésére.
- A gopher építés csapatmunka: egyrészt szinte mindig egy kisebb helyi munkacsoport tölti fel a rendszert információkkal, másrészt minden nyilvános szerver része a hatalmas „Gopher-világnak" és más gopher szerkesztők felhasználhatják a mi rendszerünket, az ott talált anyagokat és funkciókat. Ha betartjuk az írott és íratlan „játékszabályokat", mindenki munkáját sokkal könnyebbé és eredményesebbé tesszük.

A gopher szerver program telepítése

Honnan lehet beszerezni a gopher programokat?

A gopher rendszer fejlesztését továbbra is nagyrészt a Minnesottai Egyetemen végzik. Így a unixos szerver programok legfrissebb változatai anonymous ftp-vel a *boombox.micro.umn.edu* gépen a */pub/gopher/Unix* könyvtárban érhetők el. A gopher állományok *gopherX.XX.tar.Z* néven találhatók meg a könyvtárban. Ez az állomány forráskódban tartalmaz egy kliens és egy szerver programot, továbbá néhány hasznos segédprogramot is.

Ezt az FTP archívumot természetesen el lehet érni gopheren keresztül is: a *gopher2.tc.umn.edu* gépre kell csatlakozni, majd az „Information about gopher" menüben a „Gopher software developement" könyvtárat kell kiválasztani.

A VMS-hez készült gopher szervert teljesen külön fejlesztik. Ennek állományai a *trln.lib.unc.edu* gépen érhetők el a gopher segítségével.

(Ugyanakkor a VMS-es gopher klienst a unixos csomagban lehet megtalálni, ugyanis a kettőnek közös forráskódja van, VMS felhasználók számára mellékelnek egy make.com nevű állományt is, ami segít a lefordításban. Ha pl. UCX alá szeretnénk lefordítani a gophert, akkor csak ennyit kell mondanunk a *config.h* szükséges módosítása után: `@make ucx`. A VMS-es gopher rendszer üzembehelyezésére itt nem térek ki részletesen, aki erről a lehetőségről szeretne többet tudni, kérem forduljon hozzám bizalommal a *popovics@gopher.bme.hu* címen.)

Nem szükséges Amerikából áthozni az új változatokat, mert ezeket az archívumokat a világ számos pontján „tükrözik", így Magyarországon többek között a Budapesti Műszaki Egyetem ftp szerverén is *(ftp.bme.hu)*. Itt vannak egyéb, a gopherrel kapcsolatos állományok is.

Jogosultságok és biztonsági kérdések

Sok gépen a gopher szerver jelenti az első teljesen nyilvános, sokfunkciós információs szolgáltatást. Ezért nagyon gondosan kell megválasztani a gopher szerkesztőinek/felhasználóinak jogosultságait és kialakítani a rendszer működési környezetét, mert sokféle szempontot kell mérlegelni és figyelembe venni. Az internetes szolgáltató gépek biztonsági kérdéseiről ma már vaskos könyveket írnak, itt most csak néhány jó tanácsra és apró ötletre jut hely. A gopher szerverünk rendszergazdájával szorosan együttműködve lehet ezeket a problémákat megoldani, és folyamatosan dolgozni kell azon, hogy a gopher szolgáltatásunk minél biztonságosabb legyen, de ugyanakkor lehetőleg ne korlátozzuk a rendszer használhatóságát és ne hagyjuk kihasználatlanul a gopher változatos lehetőségeit.

Először is a rendszergazdától hasznos egy gopher adminisztrátori accountot kérni (*gmaint*, *gopherd*, *gadmin* vagy valami hasonló); célszerűen az „ő" neve alatt fut majd a szerver program, az „ő" tulajdonában vannak a konfigurációs fájlok, és neki van joga a gopher adatbázis módosításához. Emellett azok, akik még a gopherbe adatokat visznek fel, ebben a könyvtárban rendelkezhetnek egy adott területen írásjoggal, például egy „gopher group" tagság révén. A különböző operációs rendszereknél különböző lehetőségek vannak a jogosultságok kiosztására. Például az AIX-nél minden egyes állományhoz és könyvtárhoz explicit módon beállítható, hogy kik és milyen joggal férhetnek hozzá. Solaris esetében viszont nincs ilyen lehetőség és ebből pl. ilyen gondok adódnak: A szerkesztői gárda minden tagja számára írhatóknak és olvashatóknak kell lenniük a gopher könyvtáraknak, a más csoportban levők viszont ezeket egyáltalán nem láthatják. Szükség esetén nekik egy *newgrp* és egy *unmask* parancsot kell kiadniuk, hogy a szerver könyvtárban megfelelő jogosultsággal rendelkezzenek. A probléma tovább fokozódik, ha az egyes gopher-gazdák egymás területeit nem láthatják, hanem további alcsoportokat alkotnak, de a *gopherd*-nek mindent tudnia kellene írni és olvasni is. Ilyenkor minden szerkesztő számára létre kell hozni egy külön adminisztrátori csoportot, és abba fölvenni a *gopherd* usert is . . .

Sok munkát lehet megtakarítani ennek a rendszernek a helyes kigondolásával és megtervezésével, ez ugyanis sokkal egyszerűbb, mint utólag a fájlok védettségét, tulajdonosát változtatni, vagy netán egy saját névre telepített teljes gopher adatbázist megmozdítani. (*Erről van már személyes rossz tapasztalatom, nem is egy — P.P.)*

Amennyiben nyilvános gopher klienst is telepítünk, akkor az ezt használók számára létre kell hozni egy accountot, melynek jogait minél jobban korlátozni kell. VMS esetében ez egyértelműen megoldható egy ún. *CAPTIVE* accounttal, ami lehetetlenné teszi, hogy a felhasználó a belépéskor elinduló *LOGIN.COM* után saját utasításokat hajtson végre. Így – amennyiben a *LOGIN.COM*-ból indítjuk el a gopher klienst –, amikor a felhasználó kilép a gopherből egyben a rendszerből is kilép. Unix esetében erre nem létezik ilyen általános és biztos megoldás, talán az a legcélszerűbb, ha a gopher felhasználó alapértelmezett shell-jét átdefiniáljuk egy olyan programra, amely a gopher klienst indítja, illetve magára a gopher kliens programra. (Az AIX-nél ehhez az */etc/security/login.cfg* állományba fel kell venni a gopher kliens programot a *shells* rovatba.) Gondolni kell a fájlokhoz való hozzáférések korlátozására is. Potenciális veszélyt jelenthet, ha a gopher kliensnek a szerver adatterületén kívül más könyvtárakhoz is van írásjoga. Lehetőség szerint az ilyen felhasználók számára csak az interaktív bejelentkezést engedélyezzük, az egyéb dolgokat (DECNET FAL, FTP stb.) tiltsuk le. A nyilvános gopher kliens programot a megfelelő kapcsolókkal kell elindítani, hogy a felhasználó ne nyomtathasson és ne menthessen el fájlt, esetleg telnet hívást se indíthasson és ne tudjon levelet küldeni.

Ezeknek a jogoknak a meghatározása és beállítása a helyi rendszergazda dolga, aki mérlegelheti, hogy mennyire vállalja a felelősséget és a kockázatot a gopher felhasználók tevékenységéért és a gopher account alól elkövetett esetleges betörési kísérletekért.

Hogyan kell telepíteni a gopher programokat?

A *.tar.Z* állományok tömörített archívumok, átmásolásuk után helyre kell állítani a tartalmukat. Ez Unix alatt a következő utasításokkal történik:

```
uncompress xxx.tar.Z; tar -xvf  xxx.tar
```

vagy esetleg ha *gnu tar* van telepítve, akkor:

```
tar -zxvf  xxx.tar.Z
```

Ezek után valami hasonló könyvtárstruktúra jön létre:

```
/gopher2.016/  - Alapvetõ információs fájlok, Makefile.Config, conf.h..
/docs/         - Dokumentációk, man page-ek
/object/       - A kliens és szerver által közösen használt C fájlok
/gopher/       - A gopher kliens program C forráskódja
/gopherd/      - A gopher szerver program C forráskódja
```

A docs könyvtárban található legfontosabb anyagok:

```
gopher.1       - A gopher kliens összefoglaló leírása (manual page)
gopherd.8      - A gopher szerver összefoglaló leírása (manual page)
server.doc     - Egy leírás a gopher szerver működtetéséröl...
```

Az összefoglaló leírások ún. nroff nyelven íródtak. Ahhoz, hogy egy kényelmesen olvasható változatot kapjunk, adjuk ki a következő parancsokat:

```
nroff -man gopher.1 >gopher.txt
```

illetve papírra való nyomtatáshoz:

```
nroff -man gopher.1 | lpr
```

Ezután következnek az igazán érdekes dolgok: le kell fordítani a gopher szervert. Ehhez valószínűleg szüksége lesz egy helyi unixos szakértő támogatására, továbbá a program „illendő" telepítéséhez a gép rendszergazdájának közreműködésére.

Az első lépés az ún. Makefile módosítása. A gopher fejlesztői a fordításhoz szükséges konfigurációs beállításokat a ***Makefile.config*** nevű fájlban helyezték el. Az itteni beállítások nagy része módosítható a futtatás során, de célszerű már itt a legtöbb beállítást elvégezni, hogy a szervert minél kevesebb paraméterrel kelljen majd indítani.

A Makefile.config első néhány sora a program revíziós információit tartalmazza, ezt itt most nem idézem, mint ahogy számos a fájlban található, a konfigurálást elősegítő megjegyzést sem. Először a kedvenc *C fordítót* kell kiválasztani (*cc* esetleg *gcc*), és néhány vonatkozó paramétert beállítani. Főleg az SCO unixos gépeken célszerű *gcc*-t (GNU C-t) használni, ezzel sok fordítási hibától megkímélhetjük magunkat.

```
CC = gcc

#OPT=-g
OPT=-O
```

A következő elem az operációs rendszer típusára vonatkozik, ezen nem nagyon kell változtatni, hacsak nem ütközünk fordítási hibákba. Ezek a beállítások Solaris esetén voltak szükségesek:

```
# Add -DUSG          for System V
#     -DBSD          for BSD
#     -DNO_WAITPID   if you have wait3 instead of waitpid()
#     -DUSE_FLOCK    if you have flock instead of fcntl()
#                     locking
GSYSTYPE=-DUSG
```

Bizonyos rendszereken (A/UX, SCO, IRIX) a *ranlib* utasítást a *touch* paranccsal kell kiváltani

```
RANLIB = ranlib
```

Az install utasítás az adott op. rendszeren (SCO ODT és OSF/1 esetén pl. ***bsdinst***, AIX esetén meg teljes path-al ***/usr/ucb/install***)

```
INSTALL = install -c
```

A következőkben azokat a könyvtárakat kell megadni, ahova a gopher programok és egyéb fájlok kerülnek. Ezeket a beállításokat célszerű meghagyni, ez a legkényelmesebb és legáltalánosabban elterjedt ugyanis. Ha nem szeretne ezekbe a rendszerkönyvtárakba írni, kicserélheti a PREFIX-et mondjuk a saját login könyvtárára: */home/gophermn* vagy valami hasonlóra. Ebben az esetben azon belül létre kell hozni a *bin,* az *etc* és a *lib* könyvtárakat...

```
#-----------------------------------------------
# Where shall we install stuff?
#
PREFIX              = /usr/local
CLIENTDIR           = $(PREFIX)/bin
CLIENTLIB           = $(PREFIX)/lib
SERVERDIR           = $(PREFIX)/etc
# On SCO manuals are in /usr/man but its easiest to do a
# symbolic link from /usr/local/man to /usr/man for this
# and other packages
MAN1DIR             = $(PREFIX)/man/man1
MAN5DIR             = $(PREFIX)/man/man5
MAN8DIR             = $(PREFIX)/man/man8
```

A szerver méretét csökkenthetjük, ha a nyomkövetési lehetőséget nem engedélyezzük. Ez főleg akkor hasznos, ha a szerver **inetd** alól fut, ugyanis akkor minden egyes hívás során külön be kell tölteni a memóriába. Teszteléshez célszerű egy külön („debuggolható") futtatható változatot készíteni a következő beállítással:

```
DEBUGGING = -DDEBUGGING
```

A szerver méretét tovább csökkenthetjük, ha kihagyjuk a számunkra fölösleges funkciókat:

- **DADD_DATE_AND_TIME** : Dátum és keletkezési idő megjelentetése a gopher tételek címénél
- **DLOADRESTRICT** : Akkor szükséges, ha szeretné a maximálisan kiszolgált csatlakozások számát meghatározni
- **DBIO** : A *WAIS* indexelő mechanizmus egy kiterjesztése, melynek használata csak ritka esetekben indokolt. Fontos, hogy ugyanezt az opciót a WAIS fordításánál is ugyanúgy kell megadni, mint itt.
- **DDL** : A Tim Cook által fejlesztett *dl* adatbázisok használatát teszi lehetővé
- **DUMNDES** : Az *Admit1* biztonsági protokoll kiterjesztés használatához szükséges
- **DCAPFILES** : A régi, ma már alig használt *.cap* rendszer (ld. később) használata. (Csak a korábbi adatstruktúrákkal való kompatibilitás miatt lehet szükséges)
- **DSETPROCTITLE** : BSD féle rendszereknél lehetőséget ad a processz nevének definiálására

```
SERVEROPTS = -DSETPROCTITLE -DCAPFILES -DDL -DLOADRESTRICT
```

Hasonló módon a kliens programnál is meglehet adni, hogy mi mindent fordítson bele a rendszer:

- **DNOMAIL** : nem engedi, hogy a kliensből levelet küldjenek (nyilvános kliens esetében célszerű beállítani)
- **DAUTOEXITONU** : Ha a főmenüben U-t üt a felhasználó, kilép a kliens programból

```
CLIENTOPTS = #-DNOMAIL -DREMOTEUSER -DCLIENT_LOGGER
```

A kliens és szerver library-k. Ezeknek a kiválasztásához – ha nem sikerül az alábbi lista alapján – kérjen segítséget egy helyi Unix szakértőtől.

```
#-----------------------------------------------------
# Libraries for clients and servers
#    Ultrix needs -lcursesX instead of -lcurses
#
#-----------------------------------------------------
# Libraries... Uncomment out SEQLIBS  if compiling on sequent Dynix,
#                  "       "  PTXLIBS  if compiling on sequent Dynix/ptx,
#                  "       "  UMAXLIBS if compiling under UMAX,
#                  "       "  SCOLIBS  if compiling under SCO Unix.
#                  "       "  AUXLIBS  if compiling under A/UX
#                  "       "  INTERACTIVELIBS if compiling under Interactive
#
# Note: SCOLIBS needs -lintl if using gcc to compile in
# order to find strftime
SVR4LIBS    = -lsocket -lnsl
```

...

A következő sor beállítása attól függ, hogy az ön gépén a *hostname* utasítás mit ír ki. Ha a gép neve után a teljes domain nevet kiírja, hagyja ezt a sort üresen. Ha csak a gép nevét adja vissza (pl. goliat), akkor írja be a hátramaradó részét a teljes domain névnek. Például:

```
DOMAIN        = .eik.bme.hu
```

A **SERVERDIR** a Unix fájlrendszerének azt a pontját jelöli, ahol a szolgáltatandó adatok találhatók, a **SERVERPORT** meg azt a TCP/IP portot, ahol a gopher szerver a kapcsolatfelvételre vár. Ez többnyire a 70-es port, ami a védett portok között van, csak *superuser* jogosultsággal futó program foglalhatja le. Teszt feladatokra egy, az adott gépen más célra nem használt és 1024-nél nagyobb (pl. a 7000-es) portot célszerű használni. Mindkét beállítást felül lehet bírálni a szerver indításakor a parancssorban.

```
SERVERDATA  = /d1/gopherd/data
SERVERPORT  = 70
```

A következő sorok szintén az operációs rendszertől függenek. Nem nagyon kell általában hozzányúlni.

```
#-----------------------------------------------------------
# Compatibility defines
#
COMPAT = # -DNOSTRSTR # -DNO_STRDUP # -DNO_BZERO # -DNO_TMPNAM # -DNO_VFORK
```

Ami ezután következik, azon nem kell semmit változtatni.

Ha a Makefile.config behangolása sikerült és módosította a szükséges sorokat, következik a *conf.h* fájl, ami már nem a fordítással kapcsolatos információkat, hanem a szerver és a kliens működését befolyásoló adatokat tartalmazza.

Elsőnek a kliens program beállításai közül két default (alapértelmezés szerinti) szervert adhatunk meg. Ha a klienst egyszerűen csak a *gopher* utasítással indítjuk, ezen szolgáltatók valamelyikére próbál majd csatlakozni.

```
#define CLIENT1_HOST "gopher.bme.hu"
#define CLIENT2_HOST "gopher2.tc.umn.edu"

#define CLIENT1_PORT 70
#define CLIENT2_PORT 70
```

A kliens által használt ftp gateway: állítsa egy olyan – lehetőség szerint közeli – gopher szerverre, amelyen implementálva van ez a funkció. Erre a szerverre a kliens akkor csatlakozik, ha a felhasználó az „f" billentyű megnyomásával direkt ftp kapcsolatot kezdeményez. Ha a szervert is installálja, állíthatja az ön saját szerverére is.

```
#define AFTP_HOST "gopher.bme.hu"
#define AFTP_PORT 70
```

Ha azt akarjuk, hogy a Delete utasítás csak a könyvjelzők (bookmarks) esetében működjön, akkor írjuk be a `#define DELETE_BOOKMARKS_ONLY` sort. Ezzel megakadályozható, hogy a felhasználók egy gopher menüből kitörölhessenek bizonyos címeket. (Nem a szerver adatbázisából, csak a megjelenített listából!)

A következő lehetőség, hogy a nyilvános gopher kliensek felhasználói csak a *system rc* fájlt használhatják (olvashatják) mint konfigurációs állományt. Egyéb felhasználóknak a gopher kliens első futtatásakor saját könyvtárukban létrejön ez a fájl, ami a felhasználó saját beállításait, konfigurációját tartalmazza.

```
/* #define SECURE_MAPS_GLOBALRC_ONLY /* */
```

Itt különböző platformok speciális beállításait lehet megtenni:

- a play utasítás hívásának módja, ha van
- a nyomtató (lp-lpr) utasítás

```
#if defined(sun)
#define PLAY_COMMAND "play -v 40 -"
#endif
...
```

VMS felhasználók a fájlok tárolási módját választhatják ki. (fix hosszúságú rekordok, illetve *LF*-el elválasztott rekordok). Az ezután következő beállítások VMS-re vonatkoznak csak.

Unix felhasználók két oldalt lapozhatnak. ☞

```
#if defined(VMS)
...
#define VMSRecords /* */
```

A szövegek megjelenítésére használt pager utasítást lehet kiválasztani. Véleményem szerint nem indokolt a *TPU* használata, csak lassabbá, nehézkesebbé teszi a program használatát. Léteznek azonban a „*type/page*"-en kívül más szövegmegjelenítő programok is, mint például a *MOST*.

```
#define PAGER_COMMAND "builtin" /* */
/* #define PAGER_COMMAND
"TPU/NOINI/COM=GopherP_Dir:GOPHER.TPU %s" /* */
/* #define PAGER_COMMAND "most -n +s %s" /* */
```

A következőkben a levelezést lehet konfigurálni. A *mail* utasításon értelemszerűen nem kell változtatni, annál inkább a cím képzésén. Válassza ki az ön gépén használt levelezőrendszernek megfelelő sort, és az első **MAIL ADRS** sort „kommentelje ki".

```
#define MAIL_COMMAND "mail"
/* #define MAIL_ADRS "%s"  */
/* #define MAIL_ADRS "\"IN%%\"\"%s\"\"\""   /* */
#define MAIL_ADRS "\"MX%%\"\"%s\"\"\""   /* */
/* #define MAIL_ADRS "\"WINS%%\"\"%s\"\"\"" /* */
/* #define MAIL_ADRS "\"SMTP%%\"\"%s\"\"\"" /* */
```

Amennyiben az ön gépén *Multinetet* használnak az internetes kommunikációra, állítsa be a megfelelő utasításokat és írja be a „define MULTINET" sort, ellenkező esetben (*UCX* vagy más) valószínűleg megfelelnek a megadott utasítások.

Ha a *telnet* utasítás szimbólumhoz vagy logikai névhez kötődik, gondolja meg, hogy ezek a definíciók a megfelelő szintű táblázatokban vannak-e benne (érvényesek-e a rendszer összes felhasználójára).

```
#if defined(MULTINET)
#   define TELNET_COMMAND "multinet telnet"
#   define TN3270_COMMAND "multinet telnet/tn3270"
#else
#   define TELNET_COMMAND "telnet"
#   define TN3270_COMMAND "tn3270"
#endif
```

A VMS nyomtatási mechanizmusából adódóan, a gopher kliensből való nyomtatáshoz célszerű egy külön kis parancs fájlt rendszeresíteni, ugyanis a gopher kliens a letöltött fájlokat csak a megjelenítés időtartamára tárolja, és ha a nyomtató éppen foglalt, a fájl neve ugyan szerepel a nyomtatási sorban, de maga a fájl talán már nem lesz elérhető. Ez a kis program a kliens által készített ideiglenes fájlokat lemásolja a ***sys$scratch*** könyvtárba, és onnan nyomtatja ki. Ezt a programot a rendszer felhasználói számára elérhetővé és futtathatóvá kell tenni, és ebben a fájlban a második sorban lévő nyomtatási utasítást kell kiválasztani.

```
#define PRINTER_COMMAND "print %s" /* */
/* #define PRINTER_COMMAND "@GopherP_Dir:GOPHERPRINT %s" /* */
```

Hangokat általában nem kezel a VMS. Ezeket a fájlokat csak kimenteni (vagy letölteni lehet (Kermit)).

```
#define PLAY_COMMAND "- none -"
```

Ha gépének vannak grafikus lehetőségei (DecWindows), a képek megjelenítésére használhatja az XV programot.

```
#define IMAGE_COMMAND "xv %s" /* */
/* #define IMAGE_COMMAND "- none -" /* */
```

Mivel a beépített HTML olvasó még nem működik (lehet hogy a füzet megjelenése idejére már készen lesz), csak külső programokat használhatunk. A *lynx* egy teljes képernyős, míg a *WWW* egy egyszerű sormódú HTML olvasó és WWW kliens program.

```
#define HTML_COMMAND "- none -"
/* #define HTML_COMMAND "lynx -force_html %s" /* lynx 2.2 or greater*/
/* #define HTML_COMMAND "www"                 /* WWW Line-Mode client */
```

A következőkben lehet beállítani a konfigurációs fájlok nevét és helyét...

```
#define GLOBALRC "GopherP_Dir:gopher.rc"
#define REMOTERC "GopherP_Dir:gopherremote.rc"
```

... és a gopher help fájl elérési útvonalát.

```
/*
•     Point this to the on-line Gopher+ help file.
#define GOPHERHELP "GopherP_Dir:gopher.hlp"
```

Végül beállíthatja a **NOMAIL** funkciót, ha meg szeretné vonni a nyilvános kliens program felhasználóitól a levél küldésének lehetőségét:

```
/* #define NOMAIL /* */
```

Itt érnek véget a VMS-es beállítások. Érdemes megfigyelni, hogy a VMS-es fejlesztők milyen szószátyárokká váltak ... :-)

☞ A következő beállítások ismét általánosak, természetesen csak akkor kell beállítani őket, ha eddig ez nem történt volna meg. Sorrendben: ***pager, mail, telnet, tn3270, print***, a hangokat lejátszó ***play***, a **MIME**-os dokumentumok megjelenítéséhez használt ***metamail, xv***, és végül a **HTML**-t megjelenítő programok elérési útvonalát, nevét, illetve hívásuk módját lehet megadni. A legvégén ismét a gopher konfigurációs fájlok következnek.

```
#define PAGER_COMMAND "builtin"
#define MAIL_COMMAND "/usr/local/bin/elm"
#define TELNET_COMMAND "telnet"
#define TN3270_COMMAND "tn3270"
#define PRINTER_COMMAND "lpr"
#define PLAY_COMMAND "/bin/false"
#define MIME_COMMAND "metamail -P"
#define IMAGE_COMMAND "xv %s"
#define HTML_COMMAND "- none -"
#define REMOTERC "/usr/local/lib/gopherrc.remote"
```

A gopher szerver default konfigurációjának meghatározása következik. Először a ***WaisIndex*** által visszaadható maximális találatok számát kell megadni, majd a szerver által egyszerre kiszolgálható kérések számát. Ezután a *signal()* hívás visszatérési értékének típusát kell meghatározni – ezt általában nem kell módosítani.

```
#define WAISMAXHITS 40
#define MAXLOAD 10.0
#define SIGRETTYPE void
```

Lehetőség van a hálózatra való várakozás idejének maximumát is definiálni (másodpercekben),

```
#define READTIMEOUT (1 * 60)
```

és végül a szerver default konfigurációs fájljának helyzetét:

```
#if !defined(CONF_FILE)
#  define CONF_FILE     "/usr/local/etc/gopherd.conf"
#endif
```

A gopher szerver elindítása

A gopher szerver (gopherd) program az alábbi opciókkal indítható:

```
gopherd [-DCIc][-o optionfile][-L loadavg][-l logfile]
        [-u userid][-U uid] Datadir Port
```

A **Datadir** egy könyvtár, ahol a szolgáltatandó információk találhatók. A **Port** az a TCP/IP port, amelyet a szerver figyel és ahol várja a kliensektől érkező kérdéseket.

A futtatható állomány nevének megváltoztatásával a *gopherd* program más-más funkcióját érhetjük el. Ha ***gopherls***-ként futtatjuk, a program kilistázza a Datadir-t a képernyőre, feldolgozva az összes link-et és .cap-et (ld. később). ***gindexd***-ként futtatva a program egy gopher kompatibilis index szerverként működik, a kliens programoktól érkező 7-es típusú kérésekre válaszol a protokoll által meghatározott

formában. A *gindexd* használata nem ajánlott, mivel egy külön processzt futtat és csak a korábbi változatokkal való kompatibilitás megőrzése miatt van rá szükség, ugyanis ezt a funkciót ma a gopher szerver látja el.

- **-D** Debug kimenet engedélyezése. Hasznos lehet, ha nem tudni, mi is a baj, miért nem megy valami. Használatához célszerű némileg a szerver kódját és működését ismerni.

- **-L** Ha a szerver a *LOADRESTRICT* kapcsolóval lett lefordítva, akkor ezzel a paraméterrel meg lehet határozni a maximális csatlakozások számát.

- **-l** A logfájl engedélyezése, amelybe a szerver naplózza az időt, a host-ot és a letöltött könyvtárat (egész pontosan a selector stringet) minden egyes kliens programtól érkezett kérdés után. A logfájl szintén hasznos információkkal szolgálhat, ha valami nem működik. Ha nincs *-l* kapcsoló, nincs naplózás.

- **-I** Ha a szervert nem közvetlenül, hanem az *inetd* alól indítjuk, akkor kell ezt a kapcsolót megadni. Ez esetben nem indul el a szerver démon, csupán egyetlen kérést kezel le a standard bemenetről és befejezi működését. Ezért használható a szerverünk tesztelésére is, a következő formában:

  ```
  gopherd -cCI /usr/local/etc/gopher-data
  ```

  Ha az elindítás után beírjuk a *selector*-t, a program a képernyőre írja azt a szöveget, amit az *inetd* segítségével egyébként a hálózatra küldene. (Az *inetd*-vel való futtatásról később még részletesen szólunk.)

- **-o** Felülbírálhatjuk vele a *gopherd.conf* állomány elérési útvonalát és nevét, amit a szerver fordítása előtt a ***config.h*** fájlban állíthattunk be. Ha emellett a beállítás mellett szeretnénk mégis más konfigurációs fájlt használni, hasznos lehet ez a kapcsoló.

- **-C** Kikapcsolja a szerver oldali cache-elést. A cache mechanizmus a következőképpen működik: a szerver keres egy ***.cache*** fájlt az adatkönyvtárban. Ha talál egy „viszonylag újat", annak a tartalmát elküldi a kliens programnak. Ha nem talál, vagy csak régit talál, akkor újra átnézi a könyvtárat és egy új *.cache* fájlt csinál.

- **-c** Nem hívja meg a ***chroot***-ot a csatlakozások lekezelése előtt. Biztonsági okokból ugyanis a szerver nem engedi, hogy az adatkönyvtáron (data directory) kívüli fájlokat le tudjanak tölteni. Ennek az egyik eszköze a unixos *chroot* hívás, ami a felhasználó (ez esetben a gopher szerver processz) számára beállítja, hogy a teljes fájlrendszeren belül hol lássa a gyökérkönyvtárat. Ezért ennek elhagyása kis mértékben csökkentheti a rendszerbiztonságot, noha a szerver sok más módon is védekezik az adatkönyvtáron kívül eső területek olvasása ellen. Célszerű az **-u** vagy **-U** kapcsolókkal együtt használni. Ha *inetd*-vel használjuk, akkor az **-I**

kapcsolót előre kell helyezni. Nagyon hasznos, ha az adatkönyvtáron kívüli fájlokat szeretnénk belinkelni a gopher szerver könyvtárba. Például linkelhetünk néhány helyi *man page*-t, az ftp szerverünk könyvtárát, egyes felhasználók saját alkönyvtárait, akik szintén információt szolgáltatnak, stb. Így helyet és munkát takaríthatunk meg, ami az információk duplázásával elveszne. Ha a **-c** kapcsoló nélkül futtatjuk a szervert, észrevehetjük, hogy bizonyos dolgok nem úgy működnek, mint ahogy azt elvárnánk. Ez elsősorban akkor tűnik fel, ha programokat (binárisokat) futtatunk, vagy a szerver könyvtáron kívül eső fájlokat linkelünk. Soha nem szabad tehát elfelejteni, hogy a Unix bináris állományok – olyanok mint az *sh*, a *grep, sed* és az *awk* –, melyek elég gyakran használatosak különféle kereső mechanizmusokban és „valós idejű" gopher elemekben, az ***/usr/bin*** illetve az ***/usr/local/bin*** könyvtárakban találhatók. Ha ez a könyvtár nem látszik, ezek az „utasítások" sem működnek. Ezek linkelése nem megoldás, sem egy ugyanazon disk-en lévő másik könyvtárba való fizikai linkelés, sem a visszacsatolt fájlrendszer alkalmazása nem járható út. Ha valaki mégis ragaszkodik a *chroot*-hoz, ám tegye: másolja át az összes szükséges állományt a szerver könyvtárába, mondjuk egy .bin nevű alkönyvtárba, és ezt adja meg a *PATH*-ban is. Rá fog döbbenni, hogy szerény kis programja mennyi unixos binárist igényel.

- -u **Userid** felhasználóként való futtatás. A szerver a felhasználóra vonatkozó engedélyekkel fut, ami biztosítja, hogy csak a publikusan, illetve a felhasználó által látható (olvasható) fájlok legyenek letölthetők a szerverről. Ez az opció a **-c** kapcsolóval és anélkül is használható. Például: ha egy gopher szervert egy anonymous ftp directory-n futtatunk, akkor az ***-u ftp*** opcióval érdemes futtatni, hogy az *anonymous ftp*-nek megfelelő jogosultságokkal lássák az adatkönyvtárat a gopher felhasználói is.
- **-U** Az **-u** -hoz hasonló, de az *uid*-t numerikusan kell megadni.

Mint már említettem, a gopher szerver két üzemmódban futhat: folyamatosan aktív szerver démonként, vagy *inetd* alóli szolgáltatásként. Az *inetd*-vel való futtatás előnye az, hogy ha a szerver netán elszállna, vagy elakadna, az ez esetben csak egy kapcsolat kiszolgálását érinti, nem úgy mint a szerver processznél, amit talán újra kell indítani. Arról nem is beszélve, hogy a magyar ékezetes karakterkészletek tükrözése épp az *inetd*-n alapszik. Azt javaslom tehát, hogy a kísérleti fázis után minél hamarabb térjen át *inetd*-re. Ehhez célszerű a lehető legkisebb méretű binárist (gopherd) összeállítani, a fordításnál minden fölösleges funkció kihagyásával, ugyanis azt az *inetd*-nek minden indítás során be kell töltenie a memóriába. VMS esetén is létezik az *inetd*-hez hasonló szolgáltatás, ennél azonban mindenképpen választani kell a sebesség és a biztonság között, ugyanis a VMS-en a komplikált és roppant következetes azonosítási folyamatok miatt sok időt vesz igénybe egy-egy folyamat (process) létrehozása. A VMS-hez igazán illeszkedő valódi *MultiThreads* megoldásról egyelőre nem tudok.

Tehát, ha külön szerver folyamatot szeretne és gondoskodni szeretne róla, hogy a gép esetleges újraindításakor mindig elinduljon, tegye be a *gopherd*-t indító utasítást a fent részletezett kapcsolókkal a megfelelő *startup* fájlba. Hogy ezt az adott operációs rendszer mellett éppen minek hívják, azt a rendszergazdának tudni kell, ezen fájlok módosítására úgyis csak neki van lehetősége.

Ugyanez áll az *inetd*-vel való futtatásra is: a rendszergazdának a következő sorokat kell beírnia az */etc/services* fájlba és az */etc/inetd.conf* fájlba:

```
#
# services  This file describes the various services that are
#           available from the TCP/IP subsystem.  It should be
#           consulted instead of using the numbers in the ARPA
#           include files, or, worse, just guessing them.
#
...
gopher            70/tcp                    # Gopher server
...
```

```
#
# inetd.conf This file describes the services that will be
#            available through the INETD TCP/IP super server.
#            To re-configure the running INETD process, edit
#            this file, then send the INETD process a SIGHUP
#            signal.
#
# <service_name> <sock_type> <proto> <flags> <user>
# <server_path> <args>
...
gopher  stream  tcp  nowait  gopherd  /usr/local/etc/gopherd gopherd -cCI
...
```

A fenti listákból természetesen jó pár sort kihagytam. Miután ezeket a módosításokat elvégeztük, nem szabad megfeledkezni az *inetd* újraindításáról; először egy

```
ps -ef | grep inetd
```

utasítással keressük meg az *inetd* processz PID-jét (azonosító számát). (Linuxon és még néhány rendszeren ez *ps -aux*). Majd indítsuk újra egy *SIGHUP* jellel:

```
kill -HUP xxxxx
```

ahol **xxxxx** az előbb kiderített azonosítót jelenti. AIX-en kicsit más a helyzet, ott ugyanis minden hasonló jellegű változtatást az AIX saját rendszermenedzsment eszközével, a ***SMIT***-tel is meg lehet csinálni. De mivel ez elég körülményes, ezért érdemesebb és gyorsabb a változtatást a SMIT megkerülésével, az *inetd.conf* direkt módosításával megoldani és kiadni az

```
inetimp
```

parancsot, majd a

```
refresh -s inetd
```

utasítással átkonfigurálni a futó *inetd* alrendszert.

A gopher feltöltése adatokkal

A szerver viselkedése különféle típusú fájlokkal

Az itt megemlített beállítások a **gopherd.conf** fájl szokásos értékei. Ha módosítjuk ezt a fájlt, akkor ezen alapbeállítások is megváltoznak.

- Minden fájlt vagy alkönyvtárat, melynek neve ponttal kezdődik, vagy ***etc, usr, bin, dev, core*** a neve, nem vesz figyelembe a szerver, vagy speciálisan kezel.
- A szerver a kliens programnak az adatkönyvtárban, illetve annak valamely alkönyvtárában található fájlok neveit küldi el, amiket az azután mint menüpontokat jelenít meg.
- A fastruktúrában lévő könyvtárak gopher directory objektumként, a szövegfájlok gopher text objektumként, és minden más típusú fájl a megfelelő gopher típusként jelenik meg a menüben.
- A tömörített fájlokat a rendszer automatikusan kicsomagolja *(uncompress).*
- A mail spool fájlok könyvtárként jelennek meg, melyben az egyes menüpontok címei a *mail spool*-ban található levelek subject mezőjével (tárgyával) egyeznek meg. Ily módon megjeleníthetünk rendszerünkben például levelezési listákat, vagy egy tetszőleges levélgyűjteményt. (Nagyobb méretű archívum készítése azért nem javasolt, mert itt nem megoldható a levelek dátum szerinti csoportosítása és a keresés is nehezen biztosítható - *P.P.)*
- A futtatható *shell script*-eket a szerver futtatja, majd a program standard kimenetét (STDOUT) elküldi a kliensnek, alapértelmezés szerint plain text típussal.
- A ***.html***-re végződő fájlokat WWW hipertext fájloknak feltételezi. Amennyiben a kliens oldalon definiálva van erre a típusra egy megjelenítő program (pl. lynx), és ezt a fájlt letöltjük, a gopher automatikusan elindítja a megjelenítőt és megmutatja a fájl tartalmát. Ezek után szabadon kószálhatunk a WWW világában, a WWW kliens programból való kilépés után visszatérünk a jó öreg gopher kliensünkhöz.
- A szerver abc sorrendben jeleníti meg a menüpontokat. Különbséget tesz kis- és nagybetű között.

Hivatkozások („link”-ek)

Mint ahogy már említettem, a gopher szerver alapértelmezés szerint magát a fájl nevet küldi tovább a kliensnek az adott menüpont címeként. Ily módon az adatok karbantartása nagyon egyszerű – főleg Unix alatt, ahol tetszőleges nevű fájl létre lehet hozni. Azonban ennek a módszernek az ember nagyon hamar megtapasztalja a korlátait és nehézségeit: nehézkessé válik a fájlokra való hivatkozás, a hosszú nevek begépelése, a folytonos idézőjelezgetés, a hosszú nevű elérési útvonalakról és az ékezetes fájlnevekről nem is beszélve. Ezért természetesen van lehetőség az egyes fájloknak külön-külön nevet megadni. Ezt kétféle módon is megtehetjük: egy *link* fájl vagy egy *.cap* alkönyvtárban levő fájl segítségével.

A linkek használatával egy gopher menüben különböző helyeken tárolt és különböző típusú információforrások is összefoglalhatók; utalhatunk velük a gopher világban megtalálható más fájlokra.

Alapértelmezésben a gopher a data directory-ban lévő, ponttal kezdődő nevű állományokat link fájloknak tekinti, ha szerkezetük az alábbi formátumú. (A Unixban az *ls* utasítás a "."-tal kezdődő nevű állományokat nem listázza ki, így ezek egyfajta „rejtett” fájlok. Sokféle program hoz létre ilyeneket, leggyakrabban a felhasználó saját könyvtárában, eltárolva benne mindenféle beállítást, könyvjelzőket, listákat... Erről könnyen meggyőződhet, ha kiad egy *ls -a* utasítást a login könyvtárában.) Egy link fájl több hivatkozást is tartalmazhat. Egy link definiálásához öt sort kell beírni a fájlba, amely a dokumentum szükséges elérési adatait adja meg. Például:

```
Name=Magyarországi Gopher Szerverek
Numb=1
Type=1
Port=70
Path=1/hun/egy-szam-szolg/gopher/magyar
Host=gopher.bme.hu
```

A *Name=* sorban van az, amit a felhasználó látni fog a képernyőjén, mint menüpont címet, jelen esetben „Magyarországi Gopher Szerverek”-et.

A *Type=* megadja a dokumentum típusát. A lehetséges típusokat lásd a 2. mellékletben!

A *Path=* sor tartalmazza a „selector”-t, amivel a kliens az illető dokumentumot ki tudja választani. A selector első karaktere a gopher típus betű – illetve számjele, az utána következő szöveg pedig a kívánt fájl az adatkönyvtár gyökeréből induló teljes elérési útvonala, és neve. Ez alól két kivétel van: Az egyik ha az aktuális könyvtárban lévő fájlra hivatkozunk – ez esetben a Path egyszerűen *./fájlnév*. A második,

hogyha a linkelt objektum nem fájl. Például a telnet link esetén (8-as típus) a Path sorba kell beírni azt a login nevet, melyet a kliensnek meg kell jeleníteni, mint ajánlott login nevet, abban a figyelmeztető ablakban, amit a telnet indítása előtt ír ki a felhasználónak.

A *Numb=* adja meg a bejegyzések megjelenési sorrendjét a listában (az abc-sorrend helyett). Ez a sor opcionális. Ha megadjuk, akkor viszont nem lehet a link bejegyzés utolsó sora.

A *Port=* és a *Host=* sorok megadják a szolgáltató gép teljes domain nevét és a portot, ahol a szerver működik. Ez a két paraméter helyettesíthető + jellel. A szerver az aktuális host nevét és az aktuális portot teszi be, ha +-t talál e két sorban.

Az eredeti nevek átírása a .cap directory segítségével

Ha CAPFILE opciót engedélyeztük *Makefile.config*-ban a fordítás előtt, akkor a szerver egy könyvtár olvasása során keresni fog abban egy **.cap** nevű directory-t. Ha megtalálta, megvizsgálja, hogy tartalmaz-e egy ugyanolyan nevű fájlt, mint amilyen a főkönyvtárban található. Ha létezik ilyen, akkor elolvassa, és hasonlóan értelmezi mint egy link fájlt.

Tegyük fel, hogy meg akarjuk változtatni egy text fájl címét a gopherben, de a fájlnevet változatlanul akarjuk hagyni. Ekkor egy leíró fájlt kell létrehozni a **.cap** directory-ban ugyanazzal a fájlnévvel, és a következő sorokkal:

```
Numb=2
Name=Ez itt a szép hosszú új név
```

Ezzel ekvivalens megoldás (például) egy *.names* link fájl létrehozása ugyanabban a könyvtárban, amelyben a fájl szerepel. Ha a fájl neve „szep.txt", akkor a *.names* fájl a következő:

```
Numb=2
Name=Ez itt egy gyönyörű anyag
Path=./szep.txt
```

Linkek létrehozása

Egy külső szolgáltatás belinkelése pl. a következő módon történhet:

```
Name=Kitűnõ ftp könyvtár
Type=1
Path=ftp:hostname@/path/
Host=+
Port=+
```

```
Name=Szenzációs, FTP -vel elérhetõ fájl
Type=0
Path=ftp:hostname@/path/file
Host=+
Port=+
```

Ez lehetővé teszi egy anonymous ftp könyvtár illetve fájl bekötését a gopherbe. Léteznie kell egy **/tmp** könyvtárnak az áthozott fájlok átmeneti tárolására. Ha **-c** kapcsoló nélkül futtatunk, akkor létre kell hozni egy **tmp** könyvtárat a gopher adatkönyvtárban is.

Fontosnak tartom megemlíteni, hogy a WWW-vel szemben az „ftp kliens" szerepét itt nem a felhasználónál működő kliens program látja el. Itt a szerverbe van beépítve az anonymous FTP gateway, ami – ha a fentiekhez hasonló szelektor érkezik – működésbe lép, és áttölti a gopher szerver *tmp* könyvtárába a kívánt adatokat, majd azokat gopher protokoll szerint továbbítja a kliens felé. A gateway szerepét a ***Host*** és a ***Port*** által definiált szerver fogja betölteni. Ezért szükséges újabb klienseknél beállítani egy default ftp gateway gépet, amit a kliens akkor használ, ha a felhasználó (általában az „f" gombbal) egy anonymous ftp könyvtárat nyit meg.

A finger (gépek felhasználóinak lekérdezése) is könnyen hozzáadható a gopher szerverhez a következőképpen:

```
Name=A HELKA jelenlegi felhasználói
Type=0
Path=
Host=helka.iif.hu
Port=79
```

Ha egy adott felhasználóra vagyunk kíváncsiak:

```
Name=Információk rólam
Type=0
Path=popovics
Host=helka.iif.hu
Port=79
```

Ez a kis játék igen könnyen megérthető, ha összevetjük a finger és a gopher protokollt.

Programok futtatása

A gopherből programok is indíthatók, például „burokprogramok" (shell scriptek) vagy ***perl*** nyelvű programok. Egy programra így hivatkozhatunk a gopherben:

```
Name=Ez itt egy okos kis program
Type={egy típus}
Path=exec:"args":/programnév
Host=+
Path=+
```

Itt az „args" helyén kell megadni a programnak átadandó paramétereket, majd a program elérési útvonalát és nevét a gopher gyökérkönyvtárhoz képest. Általában ezt használják más típusú gateway scriptek esetén is. Például az első script kereshet és visszaadhat néhány találatot. Majd „exec" scripteket generálhat, amik letöltik azt a dokumentumot, amelyet talált.

A shell scriptnek a *#!* karakterrel kell kezdődnie, továbbá futtathatónak kell lennie. A Unixban, ha egy fájl a *#!/bin/sh* szöveggel kezdődik, az azt jelenti, hogy a következő sortól kezdődő szöveget a */bin/sh* nevű program (shell) bemenetére adja – magyarán szólva a következő kódot a shellel értelmeztesse.

A program kimenetének meg kell felelnie a *Type* mezőben megadott típusnak, hogy a kliens program megfelelően meg tudja jeleníteni. Így nemcsak egyszerű szöveges dokumentumokat hozhatunk létre, hanem például listázhatunk könyvtárakat, vagy ha valakinek ahhoz van kedve, akár hangeffektusokat vagy képeket is előállíthat a felhasználó választásainak függvényében.

A gopher adatterület egyes részeinek elrejtése

A következő trükk hasznos lehet néhány esetben, amikor a *gopherd.conf* „ignore:" címkéje nem tud elrejteni valami speciális dolgot, ezért ha nem akarjuk, hogy pl. a „teszt" fájl megjelenjen a menüben, a *.names* fájlba a következő bejegyzést kell tenni:

```
Type=X
Path=./teszt
```

(tehát egyszerűen a típust kell „X"-re cserélni.) Ez hasznos lehet akkor ha pl. linkelés van a directory-ban, viszont nem akarjuk, hogy bizonyos címszavak megjelenjenek. Ez a megoldás nem zárja ki, hogy valaki explicit módon megadva ezt a szelektort, hozzáférjen a teszt fájl tartalmához.

Teljesszövegű indexek

A gopher szerver 0.8-as változatának megjelenésével jelentősen leegyszerűsödött a teljesszövegű indexek gopherbe illesztése. Csupán el kell helyezni az index-állományokat valahol a *gopher data* directory-ban, és hivatkozni kell rájuk egy link segítségével. Ennek módja attól függ, hogy milyen indexelő rendszert használunk. Az alábbiakban a WAIS és a NEXT típusú indexek beillesztését mutatjuk meg. Példánkban az indexek egy „proba" nevű alkönyvtárban vannak a *gopher data* directory legfelső szintjén.

WAIS link létrehozása

A „proba" directory-ra teljesszövegű keresés WAIS index esetében az alábbi sorokkal készíthető:

```
Type=7
Path=7/proba/index
```

Ez teljesszövegű indexre cseréli a típust. Egy WAIS index path formája: *7/{path}/{dbname}*.

NEXT típusú link készítése

A „proba" directory beépítéséhez a NEXT teljes szövegű index keresőbe egy indexet kell létrehozni az index directory-ba, és beleírni a következő sorokat:

```
Type=7
Path=7/proba
```

Ez teljes szövegű indexre cseréli le a típust. A NEXT index path formája: *7/{path}*.

Több indexállomány egyesítése

A gopher szerver egy index kérést szétoszthat több host között. Ez lehetővé teszi az indexek felosztását kisebb, könnyebben kezelhető darabokra.

A sokszoros keresés a következőképpen adható meg. Egy *.mindex* nevű fájl kell a szerveren, benne a következőkhöz hasonló sorokkal:

```
<host.domain|localhost> <port> <pathname>
```

Egy példa:

```
#Receptek
#
localhost 70 7/indexes/otherrecipes
ashpool.micro.umn.edu 70 7/indexes/recipes
#
#computer info
joeboy.micro.umn.edu 70 7/indexes/computer
```

A szerver ezt egy egyszerű kereshető adatként mutatja a felhasználónak. A keresés a két adatbázison való keresés eredményét adja.

Ha a „localhost"-ot adjuk meg hostként, akkor a szerver a helyi gépen fog keresni. Máskülönben a gopher démon egy másolata fog elágazni mindegyik indexre. A localhost használata hasznos, és használni kell, ha a keresett indexek a helyi lemezen vannak.

Ask blokkok

Az ask blokkok lehetővé teszik kérdőívszerű formátumok létrehozását. Ha a felhasználó kiválasztja ezt a dokumentumot, egy űrlap lesz számára látható. Amikor ezt kitöltötte, a válaszok a gopher szerverhez kerülnek vissza feldolgozásra.

Az ask blokkok két részből állnak: az első egy fájl, amely definiálja a formátumot, a második egy parancs script, amely figyeli a szerverhez visszaérkező válaszokat. Az ask blokkok mezői például a következő típusúak lehetnek (lásd még az 1. mellékletet is):

Ask : prompt	Kiteszi a képernyőre a teljes prompt szöveget 40 karakterig, és kér egy egysoros inputot.
AskP : prompt	Ugyanaz, mint az előző, de nem írja ki a képernyőre a gépelt szöveget.
AskL : prompt	Elfogad többsoros választ.
Note : text	Kiírja a képernyőre: text.
Select : prompt	Megjeleníti a promptot utána egy "check box"-szal.
Choose : prompt *choice choice choice*	Megjeleníti a promptot, és a lehetőségeket "radio button" formátumban.

Az ask blokkokhoz tartozó script

A script első sorának tartalmaznia kell a Unixban szokásos bűvös karaktereket: ***#!*** Enélkül a szerver nem veszi figyelembe a scriptet. Minden egyes, a formátumba beírt válasz a script standard bemenetére kerül, olyan sorrendben, ahogy a formátumon szerepelt. A „Select" 1-et ad vissza, ha ki van jelölve, egyébként 0-át. Ha a script tartalmaz „echo" utasítást vagy más kimenetet, a kimenet visszakerül a klienshez text fájlként.

Példa az ASK blokkok alkalmazására

A következő egyszerű példa megjelenít egy formátumot, amibe a felhasználó beleírhatja véleményét, panaszait, kérdéseit, majd ennek eredményét elküldi e-mailben a *gopherd* címre, és a felhasználónak „nyugtázza" az akció sikerességét.

Ehhez létre kell hozni például egy ***panasz.ask*** nevű fájlt, valahogy így:

```
Ask: Nev:
Ask: E-mail
Askl: Uzenet
```

Majd egy *panasz* nevű fájlt:

```
#!/usr/local/bin/perl

$mailcmd = "/usr/ucb/mail -s 'panaszlada' gopherd"; # a mail utasitas
($nev,$email,@panasz) = <STDIN>; # olvas a standard bemenetrol

open (MAIL,"| $mailcmd") || die "Cannot open $mailcmd";
print MAIL <<"EOM"; # a mail -re elkuldi az alabbi sorokat
From: $nev <$email>
--
@panasz
EOM
close (MAIL);

print <<"EOM"; # A felhasznalonak visszairja az alabbi sorokat
Tisztelt $nev <$email>!

Az Ön üzenetét a gopher továbbította a rendszergazdának.

EOM
```

Ne felejtse el a „panasz"-ra a futtatást engedélyezni:

```
chmod +x panasz
```

Még annyit jegyzek meg, hogy ez esetben a link fájlt csak a „panasz"-ra kell írni:

```
Numb=10
Name=Panaszláda
Path=./panasz
```

WWW és HTML támogatás

A gopher szolgáltatásait többnyire WWW kliensekkel is lehet használni (de pl. az ASK blokkok nem mindig működnek). Fordított irányban: a WWW formátumban, HTML nyelven írt állományok az újabb klienseknél megjelennek a menüben (a fájl típusának „h"-t kell megadni). Lehetséges a gopher adatterületen egy *.about.html* nevű HTML formátumú fájlt készíteni (pl. egy nyitóképpel), s ezt a szerver a menü lista előtt elküldi a WWW klienseknek. A gopher szerver jelenleg nem cache-eli a HTML lapokat.

A gopher kliens — nyilvános gopher kliens szolgáltatások

Abban az esetben, ha a szolgáltatás által megcélzott „olvasói körtől" nem elvárható, hogy saját gopher klienst futtasson, vagy szeretné, hogy a rendszert nyilvános elérésű terminálokról vagy PC-kről használhassák (például könyvtári olvasó), célszerű egy nyilvános gopher kliens programot telepíteni. Ebben az esetben viszont gondoskodni kell arról, hogy a *gopher* témaszámra bejelentkező felhasználók nehogy kárt tehessenek a rendszerben, illetve nehogy „névtelenül" betörési kísérleteket kövessenek el. Ilyenkor nyilvánvalóan nincs lehetőség nyomtatásra és a fájlok elmentésére sem, mivel a felhasználónak nincs lemezterülete, és a nyomtatások is nehezen találnának gazdájukra.

A kliens program szintaktikája

gopher [**-sSbDr**] [**-t** *title*] [**-p** *path*] [**-T** *type*] [**-i** *keresési feltétel*] [*hostnév*] [*port*]

A *hostnév* annak a szolgáltató gépnek a neve, amelyre a kliens csatlakozik, a *port*-ként megadott porton keresztül. Ha elhagyjuk ezeket az értékeket, a program a **conf.h**-ban megadott értékeket használja.

b - A könyvjelző oldalon indítja a klienst.
p - A megadott szelektor stringet küldi el először a szervernek.
T - Tudatja a klienssel, hogy a -p opció milyen típusú objektumra mutat.
i - Keresési feltétel (amennyiben 7-es típusú objektumra utaltunk a *path*-ban).
t - Megadja a gopher kliens számára a kezdőképernyő címét.
s - Védett mód a témaszámmal rendelkezők számára.
S - Védett mód a vendég felhasználók számára.
r - Közli a klienssel, hogy a felhasználó távoli.
D - Bekapcsolja a debug kimeneteket.

Az **s** és **S** (secure) kapcsolókkal letiltható, hogy a kliensből nyomtatni és menteni lehessen, valamint nem nyitható meg az Options menü.

Konfigurációs fájl

A kliens a konfigurációs adatokat a rendszer *gopher.rc* fájljából és a felhasználó *.gopherrc* fájljából veszi. Ezekben a fájlokban lévő opciók az O billentyű lenyomásával állíthatók a kliensben. A felhasználó természetesen csak a *.gopherrc* tartalmát írhatja át.

Mint a legtöbb amerikai fejlesztésű szoftvernél, a gophernél is gond van a nem angol nyelvű alkalmazásoknál. Mivel nincs egy általánosan elfogadott megoldás a nemzeti karakterekek használatára a gopherben, ezért országonként, sőt esetleg szolgáltatónként is más-más módon próbálják kezelni ezt a kérdést.

Kliens oldali dekódolás

Mint már említettük, a **gopher+** protokoll kifejlesztésénél az egyik szempont az volt, hogy alkalmas legyen a letöltendő objektumról, annak típusa mellett (MIME) a „nyelvét" is megadni. Amellett, hogy így a felhasználó választhat az egyes dokumentumok különböző nyelvű változatai közül, ennek a megoldásnak a másik haszna az, hogy az intelligens (és egyre intelligensebb) kliens programok ezek alapján választhatják ki a szöveg megjelenítéséhez szükséges kódtáblát, illetve dekódoló mechanizmust, és elvégzik a szükséges átalakításokat a gopher interaktív elemeinek szánt bemeneteken is. Ezt a fajta megoldást nevezzük *kliens oldali dekódolásnak*. Sajnos ez nem használható a gopher magyarítására, ugyanis mivel Amerikában fejlesztették (nyugat-európai közreműködéssel), a gopher+ a nyelvek jelölését az ANSI szabvány szerint végzi, amiből érthető okok miatt hiányzik a „Hu_HU", vagy valami hasonló nyelv. Elképzelhető ugyan, hogy kliens oldalon elvégezhetők az ehhez szükséges módosítások, és egyes szállítók például már olyan Unix rendszereket forgalmaznak, amelyek az ISO nyomait viselik magukon, de sajnos szerény vigasz ez annak, aki arra számít, hogy szolgáltatását a világ minden országából, és így a legkülönbözőbb típusú kliensekkel olvassák. Továbbá, a gopher+ által javasolt bővítéseket nem is végezték el minden kliensen, és ahol azt meg is tették, a nyelv – és így a karakterkészlet – választásának lehetőségét a szövegfájlokra korlátozták, és nem adtak megoldást a menükben megjelenő ékezetes szövegekre. Elképzelhető az is, hogy egy dokumentum noha angol nyelvű, de tartalmaz magyar ékezetes karaktereket. A megoldást még nehezebbé teszik a nyilvános elérésű kliensek, vagy más, terminál emulátorról elért nagygépes kliensek. Ez esetben ugyanis nem a gép „virtuális termináljához" kell illeszteni a kódtáblát, hanem a tényleges megjelenítő eszközhöz. Mindez elvileg megoldható, de sajnos az elterjedt gopher kliensek nem biztosítanak még karakterkészlet rá- és áthangolási opciókat.

Szerver oldali konverzió

Miután fölvázoltuk ezt a csöppet sem rózsás helyzetet, ideje rátérni a ma Magyarországon leggyakrabban használt megoldásra: ez a szerver oldali konverzió. Ennek lényege, hogy több szerver program szolgáltatja ugyanazokat a dokumentumokat, de különböző karakterkészlettel, és a felhasználó egy gopher menüből választhatja ki a számára megfelelőt. A kliens a karakterkészlet kiválasztásakor tulajdonképpen egy szervert, ill. szerver portot választ. Ezután, az adott szerver menüjében barangolva, az összes fájl és könyvtár a kiválasztott kódkészlettel kerül letöltésre.

Természetesen az nem megoldás, hogy minden információt három vagy négy formában tárolunk; ez nem csupán a tárolókapacitással való takarékoskodás miatt, hanem a többszörös szerkesztési munka elkerülése végett is elkerülendő. Megoldható a probléma oly módon, hogy egyetlen menüstruktúrát szerkesztünk, és utána valamilyen cseles programmal ez alapján elkészítjük az ennek megfelelő más kódkiosztású adatbázist is. Ebbe bele lehet azt is kombinálni, hogy az „árnyékstruktúrák" ne valódi fájlokat, csak linkeket tartalmazzanak, és a szövegfájlok helyett is csak egy konvertáló *exec* scriptre való hivatkozás szerepeljen, de még így is sokat bonyolítunk az életünkön, főleg ha interaktív alkalmazásokat is tervezünk a gopher alá.

Az igazi megoldás Kolb Zoltántól (kolb@inf.u-szeged.hu) származik, aki a gopher szerver *inetd* alatt futó változatát felhasználva, online konvertáló (ill. tükröző) eszközt készített: ez a *gmirr*.

Gopher a tükörben – gmirr

A *gmirr* program az ISO 8859-2 kódkiosztás szerint tárolt gopher struktúra tükrözését végzi PC852, CWI, repülő ékezetes és ékezet nélküli formára. A *gmirr* támogatja a gopher+ legtöbb lehetőségét, és kipróbáltan jól működik Solarison és AIX-en. Az ékezetes adatbevitelre és keresésre egyelőre nem ad megoldást. A vera.inf.u-szeged.hu anonymous FTP archívumból tölthető le, a /pub/networking/ gopher alkönyvtárból.

Installálásához célszerű kicsit módosítani, majd újra fordítani a *gopherd*-t, erről a programhoz mellékelt PATCH segítségünkre van. A *gmirr* lefordításához a ***Makefile*** és a ***config.h*** fájlokat kell módosítanunk. Ha AIX-en fordítunk, a Makefile.aix-et kell átnevezni Makefile-nak.

A *gmirr*-t csak *inetd*-ből lehet futtatni, valami hasonló konfigurációval:

```
# inetd.conf
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
gopher  stream  tcp  nowait  root /usr/local/etc/gopherd gopherd -cCI
gmirr1  stream  tcp  nowait  root /usr/local/etc/gmirr gmirr -cCI
gmirr2  stream  tcp  nowait  root /usr/local/etc/gmirr gmirr -cCI
gmirr3  stream  tcp  nowait  root /usr/local/etc/gmirr gmirr -cCI
gmirr4  stream  tcp  nowait  root /usr/local/etc/gmirr gmirr -cCI
```

```
# services

gopher          70/tcp
gmirr1          7001/tcp
gmirr2          7002/tcp
gmirr3          7003/tcp
gmirr4          7004/tcp
```

Ha a conf.h fájl megfelelő sorain nem változtatunk, ez azt jelenti, hogy a 7001-től 7004-ig terjedő portokon ugyanolyan gopher szolgáltatás fut, mint a 70-en – csak a következő kódkiosztásokkal:

Port	Karakterkészlet
70	ISO-8859-2
7001	PC 852
7002	sima (ekezet nelkuli)
7003	CWI
7004	Repu:lo" e'kezetes

Magyarországi konvenciók

Az 1994 tavaszán tartott Pocok Workshopon a jelenlévő rendszergazdák között egyetértés alakult ki számos területen. Itt a megbeszélés összefoglalója következik, melyet Pásztor Miklós *(pasztor@hugbox.sztaki.hu)* készített:

1. Ábrázolási módok:

Kívánatos, hogy a szerverek a következő módokon tegyék elérhetővé a magyar nyelvű információkat:

- 8859-2 kódban
- repülő ékezetekkel
- ékezet nélkül
- 852 szerinti kódolásban

A CWI kódban való prezentálás szintén hasznos lehet, de „szorgalmi" feladat. Az alap ábrázolási mód 8859-2 legyen, vagyis a többit ebből állítsa elő a szerver. Ez hasznos akkor, ha például a dokumentumot FTP szerver is szolgáltatja.

2. Repülő ékezetek

Még itt is több szokás terjed. Kívánatos, hogy legalább a gopher szerverek gazdái egységes használatban állapodjanak meg. Elfogadtuk a HNA-t, vagyis a HuniNet/ HaNák Ajánlást a következő, Popovics Pétertől származó módosítással:

Ott, ahol az ajánlás eszképelést ír elő, az eszképelés helyett az ékezetet jelző karaktert két szóközzel fogjuk közre. Ez a módosítás az olvashatóságot segíti.

3. Keresés, bevitel kliens oldalról

A következő működési mód kívánatos: Az egyes menüpontokban elfogadjuk az olyan kódú bevitelt, mint amiben a megjelenítés történik, és minden menüpontban elfogadjuk a repülő ékezetes bevitelt. Ez azért célszerű, mert a kliens nem feltétlenül képes ékezetes bevitelre, ha képes ékezetes megjelenítésre.

Vizsgálat és további megbeszélés tárgya

- Minden új, különösen gopher+ és MIME kérdéskörben
- Az egyes ékezetes menük más szerverek ékezetes menüpontjaira mutassanak ?
- A bevitel/keresés különböző ékezetes formái
- URL - MAIL gateway

Menedzselés és propaganda

A technikai és az információszervezési gondok mellett a harmadik probléma a gophernél a munkaszervezés szokott lenni. Egy nagyobb gopher szerver információkkal való feltöltése és a szolgáltatás minőségének megőrzése mindenképpen több embert igénylő feladat, vagyis egy csapatot kell szervezni és összetartani. Ráadásul a gopher létrehozása egy intézményben rendszerint egy teljesen új jellegű feladat, aminek nincs előzménye, így nincsen hozzá munkaerő és költségvetési keret. A gopher megjelenése óta még nagyon kevés tapasztalat gyűlt össze arra vonatkozóan, hogy hogyan lehet gyorsan és hatékonyan beépíteni a korábbi tevékenységekbe, elfogadtatni a intézményi vezetéssel. Valószínűleg az a legjárhatóbb út, ha valamilyen, már ismert szolgáltatáshoz kapcsoljuk vagy hasonlítjuk, így kevésbé tűnik újdonságnak (pl. *„az intézményben már működő számítógépes információforrásokat összekapcsoló metarendszer", „elektronikus egyetemi újság", „könyvtári katalógus az Interneten levő dokumentumokhoz"* stb.).

A legjobb hasonlat talán az, hogy a gopher-csapat olyan, mint egy újság szerkesztősége. Van egy-két főszerkesztő, több szerkesztő, s vannak külső munkatársak, tudósítók. A gopher hierarchikus felépítése miatt igen alkalmas a munkamegosztásra; könnyen megoldható, hogy mindenki csak a hozzá tartozó almenüket tudja módosítani, bővíteni. Például egy CWIS jellegű, intézményi gophernél a csapatba érdemes bevenni pár embert a számítóközpontból a rendszer üzemeltetési és programozási feladataihoz; egy könyvtárost az információk rendszerezésére; az intézmény adminisztrációjának/vezetésének képviselőjét a szolgáltatás „tekintélyének" biztosítása céljából; és esetleg valakit a propaganda/public relations osztályról vagy pl. az egyetemi újság vagy rádió szerkesztőségéből (ha van ilyen). Szakmai típusú gophernél természetesen kell egy szakember az illető témakörben, a színvonal megőrzésének érdekében. Ezenkívül minél több külső munkatársat próbáljunk beszervezni, akik majd az információs anyagokat küldik, felhívják a figyelmünket a hálózaton megjelent újdonságokra vagy az elavult, nem működő menüpontokra. A legegyszerűbb ezeket a segítőket a gopher olvasói közül toborozni.

Néhány jó tanács a szerkesztés koordinálásához

- Minél hamarabb készítsünk egy rövid, írott szabályzatot, amiben rögzítjük a szolgáltatás építésének szabályait, az egyes szerkesztők feladatait és felelősségét, a távlati fejlesztési célokat. Ezt időnként célszerű átnézni, felújítani és más embereket keresni azok helyére, akik nem tartják be a lefektetett „játékszabályokat".
- Tartsunk néha (lehetőleg rendszeres időpontokban) elektronikus vagy régimódi „értekezleteket", ahol megbeszéljük a vitás kérdéseket és az aktuális feladatokat.

Próbáljunk állandó kapcsolatban maradni az országban és a világban dolgozó többi „kollégánkkal" is a hálózati és/vagy a hagyományos konferenciák segítségével a folyamatos koordináció és az újdonságok megismerése érdekében.

- Jó kapcsolatokat kell kiépíteni az intézményben működő más számítógépes információszolgáltatókkal (pl. WWW szerkesztők, ftp archívum üzemeltetők, levelező listák gazdái, adatbázisszolgáltatók), mert a gopher jól tudja integrálni az ilyen rendszereket, ha azok tulajdonosai ezt megengedik.
- A legjobban dolgozó munkatársakat időnként valahogy jutalmazni kell, illetve meg kell próbálni elérni, hogy a kísérleti időszak után legyen legalább egy „főállású" gopher szerkesztő, aki munkaideje egy részében ezért a munkáért kapja a fizetését. Bár a gopher szoftver ingyenes a „közcélú" felhasználók számára, az üzemeltetésnek és az információforrások beszerzésének vannak költségei; ezekre is el kell különíteni valami pénzkeretet.

Nagyon fontos a felelősség, a biztonság, a copyright, a „cenzúra", a személyiségi jogok stb. kérdésében valamilyen (írott) álláspont kialakítása és egy egységes gyakorlat bevezetése. Mivel itt egy tömegkommunikációs csatornáról van szó, rendkívül sok és veszélyes probléma léphet fel, amelyek a szolgáltatás létét fenyegethetik, ha nem készülünk fel rájuk előre és nem védekezünk ellenük. (Különösen „kényes kérdés" lehet egy bárki által online tölthető gopher-terület, – pl. egy hirdetőtábla – üzemeltetése. Ugyancsak vitatott a log fájl használata és nyilvánosságra hozatala, amiből kideríthető, hogy mely gépekről mely állományokat néztek meg). Ezekben a témákban nagyon sok jó tanácsot kaphatunk a gopherekkel foglalkozó kommunikációs fórumokon és el is kérhetjük (vagy letölthetjük) más gopher szerkesztők írott „szabályzatát" (policy), amikből ötleteket meríthetünk.

Hírverés

A szerkesztőség munkájának fontos része, hogy a szolgáltatást a megcélzott felhasználói körrel megismertesse, s hogy a felhasználók véleményét, igényeit megismerje („public relations" tevékenység):

- A gophert érdemes „hivatalosan" bejelenteni a helyi, az országos és az európai nyilvántartásoknál. (Az európai szervereket a gopher@ebone.net címen lehet regisztráltatni, itthon pedig a POCOK-L listán és a Hungarian Homepage gazdájánál érdemes bejelenteni). A metarendszerek üzemeltetőit meg kell kérni, hogy készítsenek linket a szerverünkre. Az érdekelt levelező listákon, hírcsoportokban és nyomtatott kiadványokban helyezzünk el egy tárgyilagos és informatív híradást a szolgáltatásunkról. Ezt időnként megismételhetjük, ha valami lényeges újdonságról tudunk beszámolni.
- A potenciális felhasználói körnek időnként tartsunk élő bemutatókat, tájékoztatókat, oktatást. Esetleg érdemes egy demo változatot vagy egy videofilmet is csinálni a szolgáltatásunkról, ami ott is bemutatható, ahol nincs hálózat.

- Figyeljük és elemezzük a gopher használatát mutató statisztikákat. Csináljunk egy online írható „ötletládát" vagy „panaszkönyvet", hogy megismerjük a gophert használók problémáit és igényeit.
- Szervezhetünk különböző játékos akciókat is, hogy a felhasználók kicsit mélyebben, „érzelmileg" is kötődjenek a szolgáltatásunkhoz. A *CWIS* és a *hobbi* típusú gophereknél mindenképpen próbáljuk meg az interaktivitás fokát növelni és ne csak passzív olvasási és letöltési lehetőséget kínáljunk (az új gopher protokollok erre már számos eszközt biztosítanak). És igyekezzünk valamilyen „személyes" jelleget, egyedi stílust adni a rendszerünknek, amivel kitűnhet a több ezer, hasonló gopher közül.

Gopher vagy WWW?

A gopher „sikeréve" 1993 volt. Mostanában viszont a World Wide Web rohamos terjedéséről érkeznek a hírek. Jogosan merül fel a kérdés, hogy aki most kezd egy internetes információszolgáltatás fejlesztésébe, melyiket válassza?

A WWW-nek multimédiás képességei miatt kétségtelenül nagy előnye van a „szerényebb" gopherrel szemben, és a jövő meghatározó hálózati felülete lesz. A legjobb megoldás például elektronikus újságok vagy prospektus-jellegű anyagok terjesztésére, barátságos interaktív alkalmazások, lekérdező felületek kialakításához. Ugyanakkor azt is érdemes figyelembe venni, hogy egy gopher szerkesztése egy nagyságrenddel kevesebb munkát igényel, mint a szépen formázott és illusztrált Web oldalak elkészítése (és annyi esztétikai érzék sem kell hozzá). Ugyancsak legalább tízszer, de néha akár százszor gyorsabban lehet navigálni egy gopher rendszerben, mint a sok formázási és grafikai elemet tartalmazó, és ezért lassabban letöltődő WWW anyagok közt. Így azoknak a felhasználóknak, akik elsősorban információt keresnek a szolgáltatásunkban, a gopher gyorsan megjelenő menüi több sikerélményt és nagyobb hatékonyságot nyújtanak. (Természetesen a Webben is lehet gyorsan mozogni a karakteres üzemmódú lynx klienssel, de akkor a WWW oldalak megtervezésébe befektetett szerkesztői munka megy veszendőbe.) Végül: néhány évig még nem lebecsülendő szempont lesz az sem, hogy a gopher szolgáltatások lényegesen kisebb forgalmat okoznak az egyre inkább „beduguló" hazai és nemzetközi vonalakon, mint a multimédia alkalmazások; és a nálunk is egyre nagyobb számban megjelenő modemes, viszonylag lassú telefonvonalon kapcsolódó felhasználóknak ma a legjobban használható internetes információforrásokat jelentik.

Ráadásul az újabb gopher és WWW rendszerek kölcsönösen „látják" egymást, szabadon „keverhetők": a Web nézegetők megmutatják a gopher menüket és a gopherek elküldik a HTML oldalakat s szükség esetén egy megjelenítő klienst is elindítanak hozzájuk. Így a címben levő kérdésre ma még a legtöbb helyen ez a helyes válasz: ***Gopher és WWW.***

Ajánlott információforrások

- **Gopher Manual Page** - Megtalálható a gopher szerver csomagban. Kibontás után a füzetben leírtak szerint nyomtatható.
- Grajek és R.K. Marone: **How to develop and maintain a Gopher?** - Online May/June 1995.
- Popovics Péter: **Bűvészkedés a Gopherrel** - Networkshop '94 előadás a Gopher+ lehetőségeiről *(IIF, Keszthely 1994.)*
- Prentiss Riddle: **GopherCon '93 - Internet Gopher Workshop and Internet Gopher Conference**
- Rich Wiggins: **A University of Minnesota Internet Gopher rendszere: a hálózati információforrások elérésének egyik eszköze** *(Wiggins és Riddle cikkének magyar fordítása a Magyar Elektronikus Könyvtár archívumaiból tölthető le, pl. a BKE és a Miskolci Egyetem Gopheréből.)*
- Számos hasonló leírásra, kézikönyvre, FAQ-ra és egyéb dokumentumgyűjteményre találhatók mutatók pl. a KFKI gopherében *(gopher.kfki.hu)*, az *Information Services/About Gopher* menüpont alatt.
- **Levelezési listák, hírcsoportok:**

  CWIS-L *(cwis-l@wuvmd.wustl.edu)* – A Campus Wide Information Systems nevű lista az intézményi szintű információs rendszerek fejlesztői számára indult. Feliratkozni a cwis-l-request@wuvmd.wustl.edu címen lehet.

  GOPHER-ANNOUNCE *(gopher-announce@boombox.micro.umn.edu)* – Új szoftverek és szerverek bejelentései. Feliratkozni az alábbi címen lehet: gopher-announce-request@boombox.micro.umn.edu

  GOPHERJEWELS *(GopherJewels@EINet.net)* – Érdekes, gopher-alapú információforrások listája. Itt lehet tippeket kapni és más gopher-gazdákat értesíteni a legújabb és legjobb gopher szolgáltatásokról Feliratkozni a listproc@EINET.NET e-mail címen lehet. A levelezőcsoport archívumai a gopher://ftp.einet.net:7447/7waissrc%3a/.GOPHERJEWELS/gopherjewels és az ftp://ftp.einet.net/pub/GOPHERJEWELS címeken találhatók.

  Gopher-News *(gopher-news@boombox.micro.umn.edu)* – Internet levelezőcsoport. A gopher fejlesztés hírei. Feliratkozni a következő címen kell: gopher-news-request@boombox.micro.umn.edu. Az archívum a minnesotai gopheren, a gopher://mudhoney.micro.umn.edu:70/11/gopher-news-archive URL címen van.

comp.infosystems.gopher – A gopherrel foglalkozó Usenet newsgroup. A legtöbb Usenet News-t terjesztő gépen megtalálható, de gopherből is olvasható, például a KFKI (gopher.kfki.hu) vagy az IIF (mars.iif.hu) gophereken.

MVSGOPHER *(MVSGOPHER@LISTS.ACS.OHIO-STATE.EDU)* – MVS rendszerű gopherek gazdái és felhasználói számára indult levelezőcsoport. A feliratkozási cím: listserv@lists.acs.ohio-state.edu

VMSgopher-L *(VMSgopher-L@trln.lib.unc.edu)* – VMS alapú gopherek fejlesztőinek és üzemeltetőinek indított lista. Feliratkozni a listserv@trln.lib.unc.edu címre küldött *sub VMSgopher-L saját_név* tartalmú levéllel lehet.

Pocok-L *(pocok-l@kkt.bme.hu*, illetve hamarosan *pocok-l@hueam.sztaki.hu)* – A magyar gopher fejlesztők és információs rendszer gazdák levelezési lisája. (A levelezőcsoport archívuma a BME Gopheren található.) Itt minden kérdést fel lehet tenni, a listát szinte minden magyar gopher-gazda olvassa...

1. melléklet: A Gopher+ protokollban szereplő attribútumok listája

Az adatok a minnesotai Gopher Team által írt *Registry of Gopher+ Attributes* című dokumentumból származnak, melyet 1993. február 16-án módosítottak utoljára. A listában szereplők mellett létezhetnek még helyileg bevezetett és használt attribútumok. Továbbá az itt felsoroltak közül nem mindegyiket használják valójában, ill. nem mindegyik kliens tudja figyelembe venni őket. A Gopher+ újabban a MIME protokoll által használt média-típusokat is támogatja. Ezek mindenkori teljes listája az *ftp.isi.edu/in-notes/iana/assignments/media-types* címről tölthető le anonymous FTP-vel.

+INFO blokk: Az eredeti gopher protokollban az egyes menütételek leírására használt adatsort (típus, név, elérési út stb.) megelőző attribútum. Ha a szervernek egy attribútum listát kell elküldenie, akkor a lista elemeit is egy-egy +INFO blokk választja el.

+ADMIN blokk: A gopher szerver adminisztrációjával kapcsolatos információs blokk. Legalább két attribútumot tartalmaz:

Admin: A szerver gazdájának neve, telefonszáma, e-mail címe stb. Az e-mail címet < és > jelek közé kell tenni, hogy a kliens automatikusan megtalálja.

Mod-Date: Az adott dokumentum utolsó módosításának időpontja, <YYYYMMDDhhmmss> formátumban (év-hónap-nap-óra-perc-másodperc).

+ABSTRACT blokk: Az illető gopher tétel tartalmának rövid ismertetése. Több sor is lehet.

+VIEWS blokk: Az adott gopher tétel alternatív megjelenítési formáinak definiálására szolgáló attribútum. Például egy egyszerű ASCII formátumú szöveg mellett elhelyezhetjük annak PostScript vagy RTF változatát is. A +VIEWS attribútumok rendszerint ilyen alakúak:

xxx/yyy zzz: <nnnK>

ahol az xxx az információforrás típusa, az *yyy* a formátuma, a *zzz* a nyelve, és az *nnn* a mérete Kbyte-ban. (Pl. a *Text/Postscript de_DE:<187K>* attribútum egy 187 Kbyte-os, PostScript formátumú, német nyelvű szövegfájlt jelent.)

Az *xxx/yyy* a következő értékeket veheti fel:

text	(sima ASCII)
text/unicode	(UNICODE)
text/postscript	(Adobe PostScript)
text/sjis	(japán)
text/jis	(japán)

text/euc	(japán)
text/gb	(japán)
text/big-5	(japán)
text/rtf	(Microsoft Rich Text Format)
text/msword	(Microsoft Word)
text/macwrite	(Macintosh Write)
text/mime	(Multi-media Internet Mail Extension)
text/tex	(TeX)
text/dvi	(Device Independent)
file	(bináris)
file/hqx	(Macintosh BinHex)
file/uuencode	(Unix uuencode)
file/tar.z	(Unix tar és compress)
file/unixcompress	(Unix compress)
file/zip	(PC-s PKZIP)
file/tar	(Unix tar)
file/zoo	(PC-s ZOO)
file/arc	(PC-s ARC)
file/lharc	(PC-s LHARC)
file/pcexe	(PC-s .EXE)
file/macbinary	(Macintosh bináris)
image/gif	(Compuserve Graphic Interchange Format)
image/jpeg	(Joint Photographic Experts Group)
image/pict	(Macintosh képformátum)
image/jfif	
image/tiff	(Tagged Image File Format)
image/pcx	(Windows képformátum)
image/pbm	
image/pgm	
image/ppm	
image/postscript	(Adobe PostScript)
image/eps	(Encapsulated PostScript)
audio/ulaw	
audio/wave	
audio/snd	
video/quicktime	
video/mpeg	
smell/ascii	(szóban leírt szagélmény)
smell/funky	("büdös" :-))
tactile/ascii	(szóban leírt tapintási inger)
tactile/touch	("érintés" :-))
terminal/telnet	
terminal/tn3270	

A *zzz* nyelv-kódokra néhány példa:

Dán	Da_DK
Flamand (Belgium)	Nl_BE
Holland	Nl_NL
Angol (Nagy-Britannia)	En_GB
Angol (USA)	En_US
Finn	Fi_FI
Francia (Belgium)	Fr_BE
Francia (Kanada)	Fr_CA
Francia (Svájc)	Fr_CH
Francia	Fr_FR
Német (Svájc)	De_CH
Német	De_DE
Görög	El_GR
Izlandi	Is_IS
Olasz	It_IT
Japán	Jp_JP
Norvég	No_NO
Portugál	Pt_PT
Spanyol	Es_ES
Svéd	Sv_SE
Török	Tr_TR

+ASK blokk: Ez az attribútum blokk egy sor kérdést tartalmazhat, melyre a kliens valami választ küldhet. A lehetséges attribútumok:

Ask: Egy egysoros kérdést és esetleg egy alapértelmezett választ tartalmaz. A klienstől kapott válasz továbbítódik a szervernek.

AskF: Egy egysoros kérdés, melyre egy fájl-névvel kell válaszolni. Ezután a szerver elküldi a kívánt állományt, amit a kliens elment.

AskP: Egy egysoros kérdés, akárcsak az Ask, de a képernyőn nem látszik a begépelt válasz.

AskL: Olyan, mint az Ask, de a válasz több sor is lehet.

Choose: Egy egysoros kérdés és néhány lehetséges válasz (TAB karakterekkel elválasztva). A felhasználó a felkínált opciók közül választhat egyet.

ChooseF: Egy egysoros kérdés, melyre egy olyan fájl nevével kell válaszolni, ami a klienset futtató gépen van. Ezután a kliens elküldi a fájlt a szervernek.

Note: Kiírja a képernyőre a megadott szöveget, de nem kér választ.

Select: Egy ki/be kapcsolható opciót kínál fel.

2. melléklet: A legfontosabb és leggyakrabban használt gopher-típusok

Nem mindegyik kliens tudja mindegyiket kezelni és néha meg sem jelenítik a számukra ismeretlen típusú dolgokat a menükben. A „Jelzés a menüben" oszlopban a karakteres Unix kliens által használt jelzések láthatók a leggyakoribb típusok esetében.

Típus	Jelzés a menüben	Leírás
0	. vagy (nincs)	"sima" szövegfájl
1	/	gopher menü (directory)
2	<CSO>	CSO "telefonkönyv" szerver
3		hiba
4	<HQX>	BinHex kódolású Macintosh fájl
5	<PC Bin>	DOS bináris fájl
6		uuencode kódolású Unix fájl
7	<?>	kereshető index
8	<TEL>	telnet kapcsolat
9	<bin>	tetszőleges bináris fájl
g	<picture>	GIF formátumú kép
h	<HTML>	HTML (WWW) formátumú hipertext
i		"inline" szövegfájl
I	<picture>	valamilyen kép
M	<MIME>	MIME formátumú multimédia fájl
P		Adobe Portable Document Format
s	<)	digitális hang
T	<3270>	IBM (3270 terminál) telnet kapcsolat
;	<movie>	digitális video
+		redundáns szerver (az eredeti másolata)
(nincs)	<??>	ASK formátumú űrlap

4669

150,—

Bibliotheca Academiae Scientiarum Hungaricae
MAGYAR TUDOMÁNYOS AKADÉMIA KÖNYVTÁRA

A NIIF Információs Füzetek sorozatban az alábbi köteteket tervezzük. A címek melletti csillag (*) jelzi, hogy mely füzet készült el.

I. sorozat

1. Rajta vagy már a hálózaton? (*)
2. Kalandozás a Gopherrel
3. Böngészés a WWW-vel
4. Keresgélés a WAIS-szel
5. Gyûjtögetés az FTP-vel
6. Kapcsolattartás e-mail útján az X.25-ön
7. Kapcsolattartás e-mail útján az Interneten
8. Vitatkozás a USENET newsgroupokban
9. Kutatás a hálózati könyvtári katalógusokban
10. Információszerzés kereskedelmi szolgáltatók adatbázisaiból
11. Beilleszkedes a hálózat virtuális világába (*)
12. A hálózat használata a kutatásban (*)
13. A hálózat használata a könyvtárakban
14. A hálózat használata az iskolákban (*)
15. A hálózat használata elektronikus publikáláshoz
16. A hálózat használata Windowsból
17. Szórakozás és játék hálózati szoftverekkel

II. sorozat

1. Hogyan csináljunk saját Gophert? (*)
2. Hogyan csináljunk saját WWW-t?
3. Hogyan csináljunk saját FTP archívumot?
4. Hogyan indítsunk saját BITNET/INTERNET levelezõcsoportot?
5. Hogyan indítsunk saját USENET newsgroupot?
6. Hogyan csináljunk saját OPAC-ot?
7. Hogyan integráljuk hálózati információs rendszereinket?
8. Hogyan védjük meg számítógépes hálózati rendszerünket?