VieGOPHER

—

A Gopher System for VM/CMS

Gerhard Gonter

June 16, 1993

Abstract

[1] The Internet Gopher Protocol provides a simple, yet very effective method for distributed document search and retrieval. This document describes usage, installation, maintainance and also some points about the implementation of a Gopher system for IBM's operating system VM/CMS, called the VieGOPHER. Both, using the client and administrating the server are discussed.

The system can be distributed freely and is dedicated to the community of cyberspace.

[1]This file is still somewhat ... er ... incomplete ... sorry ...

Contents

List of Tables

List of Figures

1 Introduction

This document describes an implementation of the Internet GOPHER Server and Client protocol for IBM's operating system VM/CMS, known as the 'Vienna' VM Gopher System or 'VieGOPHER'. There are things left out and enhancements will follow. If you find any errors or have other suggestion then don't hesitate to contact me on one of the addresses given in section G on page 50.

The system consists of a client, based on XEDIT (`GOPHERC.EXEC` together with a bunch of `.XEDIT` and `.EXEC` files) and two servers, the older one is a simple, straight forward, plain vanilla, nothing special, ordinary, single-treaded Internet Gopher Server[2] (`GOPHERD.EXEC`), the new one (`GOPHERDD.EXEC`) is just the opposite of that all... well... the most interesting feature of it is that it is multi-threaded. This means, it can handle more than one connection at the same time and, in fact, the version I currently run listens on four ports at the same time and is still able to handle a couple of connections simultanously. Since this server listens on four different ports you might suspect that one is the officially assigned gopher port (70) and the the other two ports are something else. This assumption is not completely wrong, the server also listens on port 79 and pretends to be a finger server as well on port 17 and serves out fortune cookies. The service on port 17 is also known as the Quote-of-the-Day protocol. Finger protocol (see [10]) and Quote-of-the-Day protocol (see [9]) are handled by the Gopher Server as a by-product, as you may have time to read in the code.

In order to run either the server or the client you also need a program called `REXTCPIP.MODULE` (by Ken Hornstein). You can find it on any BITNET LISTSERV that carries VM-UTIL, (e.g. DEARN). Be sure to use version 1.8.4. or newer. Section 6 gives information about modifications that improve the stability of the program. If necessary, a modified version can be obtained from me. You also need TCPIP to run either the server or the client. The new server can also employ `YWAKEUP.MODULE` to trap messages from servicing machines.

Clients running under TCPIP version 2 can take advantage of RXSOCKET instead of REXTCPIP. RXSOCKET is generally more elaborate and stable, either version 1 or can be used.

The instability of REXTCPIP is the main weaknes of the server. To overcome this problem I work along three paths:

1. I try to make the REXX code of the server as shock-proof as possible,
2. I try to correct some of the problems inherent in REXTCPIP, see section 6 for a documentation about some of the patches sofar,
3. allowing to select RXSOCKET seems to be the most promising path in the long run.

[2] not supported anymore

1.1 Features

Client:

- can work with WAIS search gateways
- convenient XEDIT interface

Server:

- multiple connections
- integrated finger server
- integrated cookie server
- interface for embedded applications

1.2 Rose installation procedure

The source code of the programs and the parts of the LaTeX documentation are stored together in NED's HYX format — NED is a MSDOS 'hypertexty' text editor. From this source three text files are generated:

- `VIEGOPHR.PRO(DUCT)` containing the program code,
- `VIEGOPHR.TEX` containing the documentation, (this document)
- `VIEG-PRO.TEX` containing the source code listing.

`VIEGOPHR.PRODUCT` needs to be processed by `ROSE.EXEC`. Rose is not part of VieGOPHER, however, it was developed for it's installation. The program can be distributed freely just as the other package.

1.2.1 Quick'n'dirty installation

Appendix E on page 39 lists all Rose prompt texts that will be presented during the installation phase.
Use this checklist to get the system off the ground:

1 ☐ Put `VIEGOPHR.PRODCUT` and `ROSE.EXEC` on any readable minidisk.

2 ☐ Prepare a R/W minidisk with filemode A with at least 2 cylinders (275 blocks with 4096 byte).

3 ☐ If you have `VIEGOPHR.FIX` around somewhere, you can place it on minidisk A, but this is not necessary.

4 ☐ Start the program generator by entering: `ROSE VIEGOPHR`

5 □ Read the options presented by Rose and answer the questions appropriately. In most cases you will want to use the predefined option. In this case simply enter code 1 or hit the ENTER key. If you want to change something enter code 3 (followed by pressing the ENTER key of course) and then enter the value that you want to use.

6 □ After prompting for all options Rose will generate all modules and invoke the postinstallation procedure `GOPHINST.EXEC`.

7 □ Fetch any missing file that the postinstallion procedure might point out. If the postinstallation procedure finds inconstencies in your installation, remove the appropriate line from `VIEGOPHR.FIX` and go back to step 4.

8 □ Put the file `VIEGOPHR.FIX` at a safe place. You might want to use it again for a fast re-installation later on.

9 □ Place `RXSOCKET.MODULE` and/or `REXTCPIP.MODULE` on minidisk A.

10 □ Done! Your minidisk A should now contain all necessary programs for the client and server.

11 □ If you don't want the server you can remove `GOPHERDD.EXEC`.

12 □ If you want to run the server it's time to look through this document (`VIEGOPHR.PS`) now to find out about index files, access groups, etc...

2 Client

2.1 Client Modules

The client is based on XEDIT and REXTCPIP or RXSOCKET. It does not require any other non-standard extensions of the operating system, like CMS-PIPES and should also work with TCP/IP release 1 (the REXTCPIP version only).

2.2 General Module Layout

2.2.1 Modules from other Sources

The files `REXTCPIP.MODULE` and `RXSOCKET.MODULE` alternatively provide the network transport system.

TCP/IP must be available to the user; the TCPIP public disk is not required unless the user also wants to use TELNET.

2.3 Installation

2.3.1 Quick Installation

Installing the VieGOPHER client quickly is fairly easy: Just get the distribution package[3], and follow the installation procedures given in the readme file (or in the introduction), get `REXTCPIP.MODULE` version 1.8.4 or `RXSOCKET.MODULE`, put everything on your minidisk, and type `GOPHER`. If everything is ok you should get the root menu of the server `gopher.wu-wien.ac.at`[4], otherwise you have a problem.

2.4 Using the Client

The gopher client is invoked by executing GOPHER. If everything is set up properly, the root menu of your default server should appear within a few seconds. Figure 1 shows a root menu. The beginning of a menu line contains an item marker, <`F`> indicates a text file and <`D`> indicates another menu or directory. Table 2 lists all item types that are supported by this client. Please note that other servers might also provide different item types, these are flagged by a marker of the form >x<. A list of other know item types is provided in table 2.

2.4.1 Programmable Function Keys

Basically, the user interface is 'pure' XEDIT with a customized PF-key setting. How the keys are set up is shown in table 1.

PF24 triggers execution of general editor commands just like the ENTER key would do during a normal XEDIT session while the ENTER key has the same function as PF11, that is displaying the selected item. Activating the `Display` function will make the Gopher client fetch the item described by the line of the menu where the cursor is currently in, the result is then displayed as a new XEDIT file. The file containing the menu is then still active, each fetched item adds another file to a so-called XEDIT file ring.

Leaving a file with the `GoBack` function (`PF3`) causes that file to be removed from the editor ring and XEDIT will select another file from the ring for display.

This kind of behaviour might lead the user to a tree-like view of the data space. In reality — whatever that term means — is the user confronted with a quite chaotic, only partially tree like data space. One of the main features of the gopher protocol is the ability to point to other servers. Thus most servers point to the central node in the world-wide gopher network, `boombox.micro.umn.edu`, this server also maintains a list of other known gopher servers. The result is a circle of servers pointing to each other, back and forth.

[3] see appendix B.2 how to obtain it

[4] Figure 1 on page 5 shows such a menu

Figure 1: Root Index of a Gopher Server

```
 GOPHTMP  TMP0     A1  V 122  Trunc=122 Size=20 Line=1 Col=1 Alt=0
====>
<D> WU-Wien Gopher Information Service: Main Menu
gopher.wu-wien.ac.at:70 <1>
<F> About Internet Gopher (gopher.wu-wien.ac.at:70)
<D> Local Info
<F> ----------------------------------------------------------------------
<D> Gophers around the house (Univ. Economics, Vienna)
<D> Gophers around the place (Austrian Universities)
<D> Gophers around the world (and other information services)
<F> ----------------------------------------------------------------------
<D> Early Music List: archives of EARLYM-L@AEARN.bitnet + rec.music.early
<D> CERRO: the Central European Regional Research Organization
<D> WAFE: Widget Athena Frontend
<D> documents
<D> services
<D> external services
<D> pointers                        (things you might also find elsewhere)
<D> life, the NET and everything
<D> The Guide!
<F> ----------------------------------------------------------------------
<D> VieGOPHER: VM/CMS Gopher Version
 1= Help      2=(altpfs)  3= GoBack  4= Disp->BMK   5= Curs->BMK   6= BMK
 7= PageUP    8= PageDN   9= Save   10=            11= Display    12= Circle
```

Table 1: Programmable Function Keys

Key	Label	Description
PF1	`Help`	activate help menu (just for XEDIT)
PF2	`(altpfs)`	assing PF 1-12 with functions of PF 13-24
PF3	`GoBack`	quit the currently displayed item
PF4	`Disp->BMK`	save a bookmark for the currently displayed item
PF5	`Curs->BMK`	save a bookmark for the menu item that is in the line where the cursor currently is
PF6	`BMK`	activate (display) bookmark file as a menu
PF7	`PageUP`	go one page back (up)
PF8	`PageDN`	go one page forward (down)
PF9	`Save`	save the item that is currently displayed
PF10		*unassigned yet, reserved ****
PF11	`Display`	display item of line where cursor is currently
PF12	`Circle`	switch to next active item
PF13	`First`	goto first screen if still active
PF14	`(basepfs)`	assign PF 1-12 with base functions
PF15	`Exit`	leave Gopher immediately
PF16		*unassigned yet, reserved ****
PF17	`Split`	split the screen
PF18	`Join`	join screens
PF19	`PageUP`	go one page back (up)
PF20	`PageDN`	go one page forward (down)
PF21	`ShowRing`	show active items in XEDIT ring
PF22		*unassigned yet, reserved ****
PF23	`Setup`	Setup some parameters for using Gopher
PF24	`Execute`	execute command
ENTER		display item of line where cursor currently is

2.4.2 File Ring

The XEDIT file ring and the insertion and deletion strategy work like Ariadne's Thread. However, the `Circle` function allows the user to activate the next file in the editor file ring without quitting the current file. The function `ShowRing` (PF20) gives you a list of items in the XEDIT ring, you can select any item by placing the cursor in the appropriate line and pressing ENTER or PF11. All in all, the XEDIT ring system is a perfect way to loose orientation in the Gopher-maze... What was it that I wanted to explain now? Ah yes, bookmarks!

2.4.3 Bookmarks (`BMK`)

After exploring Gopher Space for a while you will realize, that an enormous quantity of information is out there. But where exactly did you see that particular piece of invaluable information? You could write down it's 'coordinates'. By the way, coordinates in Gopher Space are three components that are found in a normal Gopher directory or menu: server name, port number and selector string. The item type is useful for your client to know how a particular item should be presented, the fifth component of a directory is only of interest for the user, it is the so-called 'display string' or name and should give you a clue about what you can expect from a particular item.

VieGOPHER, like most other Gopher clients, allows you to take notes about the coordinates of an item you found. This is usually called a bookmark, because it leads you back to the place where you found the information and is not a reproduction of the information itself. You can either take a bookmark for the item that is currently displayed (PF4) or you could request that the item which is in the menu line of the cursor should be transfered to the bookmark list (PF5). The list of your bookmarks is stored in a file named `GOPHER.BOOKMARK` and can be activated by pressing PF6 (the BMK function). The bookmark file is then read and the file is added to the XEDIT file ring.
Notes:

- Save the bookmark file after modifying (editing) it using the FILE command.
- Be careful about activating and editing the bookmark file and taking more bookmarks while the bookmark file is still active in the XEDIT file ring.
- Save bookmarks for WAIS search items from the menu.

2.4.4 Saving items to a file

Text items can be stored in local files for later processing or printing. This is done by pressing PF9 (save) when displaying the text item of interest. The text will be appended to the file `GOPHER.SAVEFILE`, together with a short header listing the gopher coordinates of the item. Of course, the XEDIT commands `SAVE`, `FILE`, and `PUT` can be entered on the command line as normal.

2.4.5 Telnet

VieGOPHER supports the telnet types 8 for 'normal' telnet and T for Telnet 3270. IBM's `TELNET.MODULE` is able to handle both types, so the client will call this program by default.

It may be desireable, however, to configure another program to handle type 8 or type T sessions. This can be done either at configuration time by specifying appropriate values for the ROSE-variables `CL_TELNET_8` and `CL_TELNET_T`.

The user can overide this selections by assigning the programs he desires to the global variables `TELNET-8` and `TELNET-T`, both are in the group `GOPHER`.

Please note, the ROSE-variables and the global variables must specify the complete name of the desired program. For example, a user would use VieGOPHER's SETUP function or enter the following command to setup Arty Ecock's VT100 telnet emulator for 'normal' telnet sessions:

```
GLOBALV SELECT GOPHER SETLP TELNET-8 TNVT100 EXEC
```

2.5 Supported Item Types

Table 2 list all currently supported item types. Please be aware, some servers know item types that this client will not be able to process, other item types (like `w`) are not yet standardized and might be interpreted differently by the server on one side and the client on the other side.

2.6 Global Variables

The client uses two different types of global variables, permanent ones, for storing setup parameters, in the file `LASTING.GLOBALV` and temporary ones in memory.

2.6.1 User Setup

Table 3 on page 10 lists all the global variables that are used by the client. They are in the group GOPHER in the file `LASTING.GLOBALV` and can be modified with the SETUP function built into the client (PF23: `Setup`) or by calling the program GOPHSTP.

2.6.2 temporary variables

Table 4 on page 10 lists the client's temporary global variables, the CMS program GLOBALV keeps them in main memory. These variables can not be accessed directly.

Table 2: Supported Item Types

type	marker	short	actions on activation
0	`<F>`	File	∘ retrieve ∘ display as plain text
1	`<D>`	Directory	∘ retrieve ∘ display as directory
2	`CSO`	CSO Phonebook[a]	∘ retrieve ∘ display as plain text
7	`<S>`	search	∘ prompt for search string ∘ submit search selector to server[b] ∘ display result as directory
w	`<S>`	search	∘ prompt for search string ∘ submit search selector to server ∘ display result as file
8	`<T>`	Telnet	∘ display selector string as a clue for logging in ∘ start telnet session
T	`<T>`	Telnet 3270[c]	∘ display selector string as a clue for logging in ∘ start telnet session
i	`***`	info	(no action on activation)

[a] under construction, limited support is provided

[b] compatible with WAIS gateway

[c] Type=`T` is now officially declared as 3270 telnet mode. By default, types 8 and T are handled the same way as normal telnet, but this can be configured either at installation time or as a user setup option.

Table 3: Permanent Global Variables for the Client

`HOST`	name of the GOPHER server to be accessed at startup [Default: `gopher.wu-wien.ac.at`]
`PORT`	port of the GOPHER server [Default: `70`]
`ITYPE`	item type of the first item to be fetched [Default: `1` (*directory*)]
`SEL`	selector string of the first item to be fetched [Default: *text*[a]]
`DNAME`	name of the startup menu [Default: *none*]
`BOOKMARK`	name of the bookmark file [Default: `GOPHER BOOKMARK A`]
`SAVEFILE`	name of the file where items may be saved [Default: `GOPHER SAVEFILE A`]
`INFOLEVEL`	level of verbosity, amount of function key description [Default: `5`]
`TELNET-8`	name of the telnet program for items of `Type=8` [Default: `TELNET MODULE`]
`TELNET-T`	name of the telnet program for items of `Type=T` [Default: `TELNET MODULE`]

[a]set during installation as a ROSE definition

Table 4: Temporary Global Variables for the Client

`TMPCNT`	number of the last item fetched (0..??)
`GHOST`	host, port, type and selector of the last item fetched
`GI1:`<*ty*> `GI2:`<*ty*>	directory entry of the item held in file `GOPHTMP.`<*ty*> <*ty*> is the file type of the XEDIT file in the ring
`L`<*ty*>	screen information for active item in XEDIT ring

2.7 Files

There are no files read in normal operation. For direct access without TCP/IP, the local fetching option (`CL_USE_LOCALFETCH`) must be turned on and all the files visible to the server must also be visible to the client, especially the selector file, the group file, and the root index.

2.8 Calling `GOPHFTCH.EXEC`

The heart of the VieGOPHER client, the item fetcher **`GOPHFTCH.EXEC`** is a program of it's own right and can be embedded in other applications as well. It's calling formats are:

1.	`GOPHFTCH ofn oft ofm[(fm_opt] type host port path`
2.	`GOPHFTCH ofn oft ofm[(fm_opt] FILE ifn ift ifm`
3.	`GOPHFTCH ofn oft ofm[(fm_opt] PULL`

In the first format, all Gopher coordinates are specified on the command line, the second format requires GOPHFTCH to read the coordinates from the specified file and the third format tells GOPHFTCH that the coordinates can be pulled from the stack.

`ofn oft ofm` specifies the file where the fetched data should be written to, unless `fm_opt` forces the data to be processed otherwise.

`fm_opt`, the filemode options, can be either empty, `PUSH`, `QUEUE` or `XEDIT`.

Option	Description
	The empty filemode option indicates that the data should indeed be written to the specified outout file.
`PUSH`	indicates that the data will be pushed to the stack.
`QUEUE`	indicates that the data will be queued on the stack.
`XEDIT`	indicates that incoming data should immediately be written to the file being edited with XEDIT.

Please note:

- there must be no space between the output filemode, the bracket and the actual filemode option!
- Please note also, a valid filename for writing must be specified in all cases!

Figure 2: Format of the data on the stack for options `PUSH` and `QUEUE`

`TOP`
`-`*complete text line*
`+`*beginning of a long text line*
`+`*more of the long text line*
`-`*end of the long text line*
`-`*another short text line*
`END`

Figure 3: Format of the Gopher coordinates on file for the FILE-method

type
host
port
path

type specifies the type of the item to be fetched and controls the way how GOPHFTCH will process the data.
host specifies the internet address of the Gopher server's host.
port specifies the TCP port number where the Gopher server listens.
path specifies the selector string which GOPHFTCH will send to the server to request an item.
ifn ift ifm specifies the input file where the necessary Gopher coordinates can be found.

Long path-strings can be segmented into a series of `PATH=` or `HTAP=` records. `PATH=` segments are put at the end of the path string, while `HTAP=` segments are put at the beginning of the path string. The `END` record must be present to tell GOPHFTCH when to stop reading from the stack.

Figure 4: Format of the Gopher coordinates on the stack for the PULL-method

`TYPE=`*type*
`HOST=`*host*
`PORT=`*port*
`PATH=`*path*
`HTAP=`*path*
`END`

Figure 5: ROSE Installation section in GOPHERDD.EXEC

```
/* to be configured at installation time by ROSE: - - - - - - - - - - - */
if logfile=''      then logfile=      #<LOGFILE>#;
if selectorfile='' then selectorfile= #<SELECTORFILE>#;
if groupfile=''    then groupfile=    #<GROUPFILE>#;
if rootindex=''    then rootindex=    #<ROOTINDEX>#;
if gopherport=''   then gopherport=   #<GOPHER_PORT>#;
if fingerport=''   then fingerport=   #<FINGER_PORT>#;
if cookieport=''   then cookieport=   #<COOKIE_PORT>#;
if sreport=''      then sreport=      #<SRE_PORT>#;
/* - - - - - - - - - - - - - - - - - - -  end of configuration section */
```

3 Server

3.1 Server Modules

3.1.1 Multi-Threaded Multi-Protocol Daemon

The file `GOPHERDD.EXEC` resembles the main module of the server. This program will also call the modules `FINGERXX.EXEC` to generate the response for finger queries and `COOKIE.EXEC` to generate short text messages from a cookie base.

3.1.2 Modules from other Sources

For managing TCP connections, the file `REXTCPIP.MODULE` is used, trapping control messages from servicing VMs is implemented using `YWAKEUP.MODULE` or IBM's standard WAKEUP method.

TCP/IP must be available to the server, however, access to the TCPIP public disk is not necessary.

3.2 Installation

The files `GOPHERD.EXEC`[5] as well as `GOPHERDD.EXEC` have an installation section where local parameters should be configured at installation time using ROSE. There are the definitions of the ports where the servers listen and the files that are used by the server modules. Figure 5 shows an example.

[5] This program is now obsolete; it is still distributed in the package to provide the interested reader with some in-depth information.

3.2.1 Virtual Machines

The setup used at awiwuw11.wu-wien.ac.at is this: the virtual machine **GOPHER** has all the programs on it's minidisk 191, accessed as A. This disk is public to all users. Data files are stored on various disks. The VM also has a link to GOPHERD.191 as I.

The VM **GOPHERD** has a link to GOPHER.191 as minidisk B (read-only) and all the necessary data disks of GOPHER and other VMs. The server module GOPHERDD is started from the profile[6]. GOPHERD can be forced and auto-logged by the VM TCPIP and is currently also forced and auto-logged by the VM VMUTIL four times a day.

Allowing TCPIP to control GOPHERD requires an entry in the file PROFILE TCPIP. Figure 3.2.1 on page 15 shows an excerpt of that file, listing relevant entries. Please note that GOPHERD is mentioned in the AUTOLOG section once and three times in the PORT section, where the statements for port 17 (Quote) and 79 (Finger) are flagged with NOAUTOLOG. It's not clear if it could cause problems when one virtual machine has more than one autolog-controlled ports and sofar this setup did not reveal any shortcoming.

3.3 Files

The server processes a number of files to obtain vital information for it's operation. Purpose and format of these files is described in the following sections.

`GOPHERD.GROUPS`	privileged IP addresses, access groups
`GOPHERD.DSKGRP`	list of public data disks
`GOPHERD.SELECTOR`	known selector strings
`GOPHERD.EAI`	embedded application interface
`GOPHERD.INDEX`	main (or root) index

3.3.1 GOPHERD.GROUPS

This file defines the known access group names and lists the privileged IP addresses that are contained in a group. Records have the form...

$$\boxed{\textit{group } (\text{␣}\textit{addr})+}$$

Example 1:

```
* File GOPHERD GROUPS
*
```

[6] A sample profile is shown in section A.2.

Figure 6: Excerpt from PROFILE TCPIP

```
...
...

AUTOLOG
  FTPSERVE  PASSWD     ; FTP Server
  GOPHERD   PASSWD     ; Gopher, Finger and Quote Server
  LPSERVE   PASSWD     ; LP Server

...
...

ENDAUTOLOG

PORT
  17 TCP GOPHERD   NOAUTOLOG ; Quote Server
  20 TCP FTPSERVE  NOAUTOLOG ; FTP Server
  21 TCP FTPSERVE            ; FTP Server
  23 TCP INTCLIEN            ; TELNET Server
  25 TCP SMTP                ; SMTP Server
  53 TCP NAMESRV             ; Domain Name Server
  53 UDP NAMESRV             ; Domain Name Server
  70 TCP GOPHERD             ; Gopher Server
  79 TCP GOPHERD   NOAUTOLOG ; Finger Server
 109 TCP POP2                ; Post Office Protocol Version 2

...
...
```

```
* list of know groups
gg       137.208.
wu-wien  137.208.
edvz     137.208.16.
schnurli 137.208.16.39
```

where ...

137.208.	...	is the network address of the Univ. of Economics.
137.208.16.	...	is the sub network of the computer center.
137.208.16.10	...	is the author's PC

NOTES:

- The group `*` is not defined in `GOPHERD.GROUPS`. Files refering to group names will use `*` to indicate that something is available to everyone on the Internet.
- There is a dot (`.`) after subnet addresses and that there is no dot after a completely specified host address.

3.3.2 `GOPHERD.DSKGRP`

The disk groups file associates file modes (minidisks) with access groups, as defined in the previously. This allows the server to quickly decide if files with a certain file mode can be delivered to a client, depending on the client's Internet address.
Format of a disk group file[7]:

disk priority groupnames

An entry line starting with `*` or `#` is comment and ignored.
disk is the file mode for the particular disk. Please note, `X1` is not treated the same as `X2` etc.
priority when scanning the entries during a run-down, the file with the highest priority is delivered. Only files of priorities larger than 0 are sent out at all.
groupnames is a list of group names defined in the group file (see 3.3.1).

[7]For an example see appendix A.6.

3.3.3 `GOPHERD.SELECTOR`

This file lists all known selector strings (path names) together with associated access group names, owner authentication codes, and finally the file name. The server checks the requested item in the selector list and sends out the contents of the associated file if the file has no access restrictions. Records have the form...

fn.ft.fm	*owner*	*group*	*selector*

where

fn	file name
ft	file type * indicates that the file type is taken from the selector string. This is used to reduce the number of entries in the selector file and to have less administrative overhead when installing new files.
fm	file mode
owner	a short string identifying the owner of the item * for the general public
group	a group name defined in GOPHERD.GROUPS or * for the general public group
selector	any text string used as selector

NOTES:

- Don't use characters in the selector string which are difficult to be processed in the command line.
- Group names, owner names as well as selector strings are case sensitive.
- Owner names have currently no real meaning and are ignored
- Note the dotted notation for file names.
- Include the file mode in the filename field.

3.3.4 `GOPHERD.EAI`

(info is coming up later, remind me...)

Format of the path string (selector):

`EAI-`*handle* `TAB` *key-term*

The file `GOPHERD.EAI` defines a relationship between a selector handle, a privilege group declaration and a method to call the embedded application. The format of such a file is:

handle reserved group calling sequence

The calling sequence resembles a command line for calling the embedded application. It can contain templates that will be replaced with specific parameters by the gopher server before executing the command line. These are currently defined templates:

template	replacement
`@kw`	key term as delivered in the selector
`@fn`	filename for the output file
`@ft`	filetype for the output file
`@fm`	filemode for the output file

3.3.5 `GOPHERD.INDEX`

This file is the main index (root index), it is sent out whenever an empty selector string comes in.

3.3.6 Format of an Index File

The server does not distinguish between index files and any other text file, therefore, index files are implicitly just text files that have the directory structure as defined in [1]. Thus, all records in an index file have the form...

type display `TAB` *selector* `TAB` *host* `TAB` *port*

type	...	document item type (see table 2 on page 9 for list)
display	...	string displayed in the index
selector	...	string sent to server by the client to retrieve the document
host	...	host address of the server
port	...	port number of the server

NOTES:

- The blanks in the line above are only used for readability, they will not be found in an index file.
- The `TAB` character has hex code `'05'x` in EBCDIC (`0x09` in ASCII).
- The server sends an index file (like any other file) without any filtering.

3.4 Global Variables

Global variables are used to define a couple of parameters for the server, they are set and accessed using GLOBALV and are found in the group GOPHER in `LASTING.GLOBALV`. Table 5 on page 20 lists all relevant global variables for the multi-threaded server daemon.

3.5 Hooks in the Server Code

The GOPHERDD server code provides a number of hooks. A server administrator can install programs that are called by the server under certain conditions.

3.5.1 GSH_REAC EXEC

Executed after an REACCESS event, triggered either by a SYS-REACCESS request from the gopher server or by a SYS-REACCESS message.

3.5.2 GSH_WAIT EXEC

Execute some commands after n seconds of idle waiting.

3.6 The Gopher Server in Action

- interpreting the selector string
  - `FILE-*`
  - `SPEC-*`
  - `SYS-*`
  - `EA-*`
  - selector evaluation
  - :from:to notation
- LOG-file (+ example)

4 Server Administration Support Programs

4.1 Index Editor

The Index Editor provides an interactive tool for modifying index files. The program is, as usual, set up as a XEDIT macro, the function keys are defined in table 6 on page 21. It consists of the following modules:

- `EDINDEX.EXEC`: the editor for INDEX files

- `EDINDEX.XEDIT`: screen mask for INDEX files

Table 5: Global Variables for the Server

`LOGFILE`	defines the name of the logfile file (default: `GOPHERD.LOG`)
`ROOTINDEX`	defines the name of the file with the main menu (also called root index) (see section 3.3.5 on page 18) (default: `GOPHERD.INDEX`)
`SELECTORFILE`	defines the name of the selector file (see section 3.3.3) on page 17) (default: `GOPHERD.SELECTOR`)
`GROUPFILE`	defines the name of the group definition file (see section 3.3.1 on page 14) (default: `GOPHERD.GROUPS`)
`EAIFILE`	defines the name of the embedded application handle file (see section 3.3.4 (default: `GOPHERD.EAI`)
`DSKGRPFILE`	defines the name of the disk group definition file (see section 3.3.2 (default: `GOPHERD.DSKGRP`)
`GOPHERPORT`	defines the TCP/IP port the daemon listens to perform GOPHER protocol[a] connections (default: port 70)
`FINGERPORT`	defines the TCP/IP port the daemon listens to perform FINGER protocol[b] connections (default: port 79)
`COOKIEPORT`	defines the TCP/IP port the daemon listens to perform Quote-of-the-Day protocol[c] connections (default: port 17)
`SREPORT`	defines the TCP/IP port the daemon listens to perform SRE protocol[d] connections (default: port 150)

[a] see [1]
[b] see [10]
[c] see [9]
[d] no documentation on SRE is currently available

Table 6: Programmable Function Keys in `EDINDEX`

Key	Label	Description
PF1		
PF2	`append`	append a new record and copy data
PF3	`quit`	quit the editor indexer
PF4	`clear`	erase information form current record
PF5	`insert`	insert a new record and copy data
PF6	`delete`	delete a record
PF7	`back`	display previous record
PF8	`next`	display next record
PF9		
PF10		
PF11		
PF12		

4.2 Server Scanner

The file `GOPHSCAN.EXEC` is a program that can scan a specified server for all it's menus. It can be used for testing the server, or to get a master index for that server. The program can also scan several servers in one session, for instance all servers on campus to get a master index for the complete campus.

`GOPHSCAN` will only scan those servers, that are specified on invocation, so it will **not** wade through all gopher space the minute it is started. However, even small servers turn out to have a lot of data, so it might be possible, that one run leads to an enormous amount of data.

The program was invented for providing a seperate program with data for an experimental 'on-campus Veronica'. The other program is not finished yet and will be released as a seperate package, eventually.

5 Spin-Off Programs

5.1 Remote User Information Protocol: (FINGER)

5.1.1 `FINGERXX.EXEC`

This program prepares the data for a user information query and writes it either to a file or the console. The program is normally either called by `GOPHERDD.EXEC` to process a FINGER query or a `SPEC-FINGER` request from the Gopher server. `FINGERXX.EXEC` is usually not invoked by the user directly. The implementation of the output preparation module tried to fulfill all requirements of RFC1288

[10] and it can now be said that it is compliant to this standard.
User information has several options and consists of several parts:

1. empty query string ({C} query)
   The program returns the list of all logged in virtual machines. RFC1288 specifies that a finger server "MUST at least provide the full name" of all online users. The full name of the owner of a virtual machine is looked up in the file `LOCAL.NAMES` using NAMEFIND.

2. in-depth status of a specified user ({U}{C} query)
   The program tries to return following information components:

   (a) user log-on status:
   RFC1288 requires: "An answer MUST include at least the full name of the user." The program returns the userid, full name, terminal number and the phone number (local extension), provided that this information can be found in the lookup files.

   (b) a user information file: `FINGERID.`*userid*
   The contents of this file is returned without filtering, it will reside on one of the Gopher service machine's public disks so that the GOPHERDD daemon as well as any user invoking FINGER has read access to the file.

   (c) information from a phone book file: `PHONEBK.NAMES`

3. phone book lookup: a string containing blanks
   This query type is interpreted as a full name (which is invalid as a userid under VM/CMS) that has to be looked up in the phone book. Any phone book entry matching the query string will be returned.

4. phone number query: `#`*number*
   This query type is interpreted as a phone number that has to be looked up in the phone book. Any phone book entry matching the query string will be returned.

Full names are contained in the file `LOCAL.NAMES`, it is looked up using NAMEFIND.

The following fields are required to be present	
`:nick.`	the userid of the virtual machine
`:name.`	the name of the virtual machine's owner

The phone book is a file named `PHONEBK.NAMES` which is processed using NAMEFIND.

The following fields are required to be present	
`:nick.`	any unambigous string
`:name.`	the user's full name
`:cnt.`	contents: list of all other field tags

The following fields are present at my installation	
`:phone.`	the user's phone number
`:fname.`	the user's first name

Notes:

- A `/W` token as defined in RFC1288 is ignored and a message indicating the absence of verbose information is appended to the output.
- Finger forwarding is actively refused by returning the message: "Finger forwarding service is denied" as defined in RFC1288.

5.1.2 FINGER.EXEC

This program is a spin-off product of the FINGER protocol implementation in GOPHERDD and interfaces `FINGERXX.EXEC` for local and remote user information queries. The program can contact finger servers on remote hosts and display the resulting information on the console.

5.2 Quote-of-the-Day Protocol: (COOKIE or FORTUNE)

The program `COOKIE.EXEC` is called by `GOPHERDD.EXEC` to produce a short text file used as a fortune cookie served by the Quote-of-the-Day protocol [9] server. The program can also be invoked by the user directly and displays the message at the user's console.

Two files are used, `FORTUNE.TXT` which holds the fortune texts and `FORTUNE.IDX` which lists the beginning and end record numbers of all messages in the cookie database. Any user invoking COOKIE needs read access to these files, consequently, these files will be held on the Gopher service machine's public disk.
A couple of modifications seem to be appropriate:

- different cookie bases with different type of contents might be a nice idea.

Notes:

- The server is RFC865 compliant but listens only on TCP port 17 and not on a corresponding UDP port.

6 Known and Exterminated Bugs

6.1 `REXTCPIP.MODULE`: `TCPopen()` timing

For some reason, the VM client could not talk to the VM server, if both were on the same machine. This was a problem intrinsic to the original version of REXTCPIP, it is discussed here and a fix is presented in figure 7 later on.

What happened exactly, when the orginal versions were used:

- When the client opens a TCP connection to the server, the server changes its status from *listening* at the port to *receiving*, the connection status is then '`Connected`' (Just as wanted).
- However, the client's TCPOPEN call doesn't return so the client program hangs and the server's connection status is never changing. The single-threaded server could not handle other requests during that time, for the multi-threaded server, however, this wasn't really a problem, just a bit more of overhead.
- If the client is finally killed and the TCB (TCP control block) gets removed by the system (this can be triggerd by calling NETSTAT on the client's VM) the TCP connection on the server side breaks down and the single-threaded server returns to the listen status and behaves normally again, the multi-threaded server simply eliminates that particular connection.

One solution for this problem was to allow the local client to access the servers mini disk(s) directly. Consequently, local connections with direct access to the server's files did not use TCPIP at all. I was a bit lazy implementing all system and special functions of the GOPHERDD-server in the client so some of these will not work.

This kind of behaviour was finally identfied as a waiting problem intrinsic to `REXTCPIP.MODULE`. The function `TCPopen()` waits for a *change* of the TCPIP machines control flag for the TCP connection. The flag indicates the presence of an established connection. The code assumes that a connection is not established without any noticable delay, but, in fact, the connection is established almost immediately and the control flag does not change anymore. Due to that, this wait call will result in a loop waiting endlessly. Figure 7 describes a fix for this problem where the wait macro is skipped for local connections. As of now this worked fine.

6.2 `REXTCPIP.MODULE`: `TCPclose()` hanging

`REXTCPIP` used to wait for the other side to *propperly* close the connection before the program closed the connection and control data structures itself. In some case this caused the server to hang because the client wasn't either able to close the connection or didn't implment the protocol carefully enough. A

Figure 7: `REXTCPIP.ASSEMBLE` Fix 4

```
*
* Wait for the connect notificatione
*
WOPENTOP EQU  *
         OI   COMMFLAG,WAITFLAG+CONSTFLG Get new wait flag
         CLC  BLSOCKAD,BFSOCKAD
         BE   GGA
         WAIT PSW=WAITPSW,OLDPSW=WOPSW Wait on it
*        BE GGA
* GOODWILL EQU  *
*
* Got a response ... check it
*
GGA      EQU  *
         CLC  BCONN,CONNNUM            Our connection number?
         BNE  WOPENTOP                 If not, continue on wait
         L    R3,BVADB                 Get new state
*
* Return a code based on connection status.
*
         C    R3,=A(TRYOPEN)           Is connection still watiting?
         BE   WOPENTOP                 Go wait some more
*
         C    R3,=A(NOEXIST)           Connection exist still?
         BNE  GOODCONN                 If so, we got a good connetion
```

Figure 8: REXTCPIP.ASSEMBLE Fix 2

```
*
* Wait for connection change record
*
TOPCWAIT EQU  *
         OI   COMMFLAG,WAITFLAG+CONSTFLG   Wait for conn stat update
         WAIT PSW=WAITPSW,OLDPSW=WOPSW Wait for event
*
* Return the code given in RETCODE, and get rid of connection record
*
CLOSERET EQU  *
*        CLI  DCONNSTA,CONNCLOS        Is the connection closed?
*        BE   FINISHCL                 If so, then leave
*        CLI  DCONNSTA,NOEXIST         Is the connection gone?
*        BNE  TOPCWAIT                 if so, then leave
* Waiting is nice, but it caused more troubles than it is worth (GG)
FINISHCL EQU  *
         L    R8,NEXTCONR              Get next connection record
         ST   R8,0(R10)                Remove that link
         L    R2,DLBUFANC              Get pointer to rec buffer
         DMSFRET DWORDS=MAXBYTES/8,LOC=0(R2)  Release rec buffer
         DMSFRET DWORDS=CONRLEND,LOC=0(R9)  Drop storage
```

fix in REXTCPIP.ASSEMBLE as shown in figure 8 allows the server to close the connection without waiting for the client.

6.3 REXTCPIP.MODULE: GetIPAddr() name resolution

The function GetIPAddr() in REXTCPIP.MODULE ands over most of the work to the nameserver and returns whatever he first nameserver comes up with.

The NAMESRV virtual machine might not deliver the correct IP-address as expected, sometimes the lookup would simply fail and no IP-address would be returned at all. Apparently, the function GetIPAddr() together with NAMESRV are quite 'case sensitive' so that the function returns different IP-addresses for various upper/lower case combinations of a given host name.
Examples:

- (done second week of March 1992)

Figure 9: `TCPIP.DATA`: defining the name server

```
;
; NSINTERADDR specifies the internet address of the name server.
; LOOPBACK (14.0.0.0) is the default value (your local name server).
; If a name server will not be used, then do not code an NSINTERADDR
; statement (Comment out the NSINTERADDR line below).  This will cause
; all names to be resolved via site table lookup.
;
; NSINTERADDR  14.0.0.0
NSINTERADDR  137.208.3.4
NSINTERADDR  137.208.3.5
```

`ganymed.wu-wien.ac.at`	$\Longrightarrow$	failed
`GANYMED.WU-Wien.AC.AT`	$\Longrightarrow$	137.208.5.2
`GANYMED.WU-WIEN.AC.AT`	$\Longrightarrow$	failed
`Ganymed.WU-Wien.AC.AT`	$\Longrightarrow$	failed
`uicvm.uic.edu`	$\Longrightarrow$	128.248.2.50
`uicvm.UIC.EDU`	$\Longrightarrow$	128.248.???.???

- (done third week of March 1992)

`ganymed.wu-wien.ac.at`	$\Longrightarrow$	137.208.5.2
`GANYMED.WU-Wien.AC.AT`	$\Longrightarrow$	failed
`GANYMED.WU-WIEN.AC.AT`	$\Longrightarrow$	failed
`Ganymed.WU-Wien.AC.AT`	$\Longrightarrow$	failed
`uicvm.uic.edu`	$\Longrightarrow$	failed
`UICVM.UIC.EDU`	$\Longrightarrow$	128.248.2.50

As you can see from these examples, this situation was unsatisfying and so FDNS was created to provide a quick hack to circumvent `GetIPAddr()`.

After several days of discussion[8] and testing the following solution was found: removing the entry `NSINTERADDR 14.0.0.0` from the file `TCPIP.DATA` and providing an entry pointing to a UNIX nameserver, as shown in figure 9, works reliably.

Please note also that this problem can't be solved by simply translating the host name either to lower case or to upper case!

Maybe it's possible to find even better solutions for this problem, if you do, please drop me a line. In my oppinion, improving `GetIPAddr()` is the most promising path to take.

[8] with Ken Hornstein and Alan Flavel

6.4 `viegophr.tex`: loop

The list of known bugs in `viegophr.tex` contains a non-terminating loop. (See [4], section 6.4.)

7 (misc)

7.1 Wishes

- Please report any bugs you find and feel free to make suggestions for improvements etc...
- Send me email if you've installed either the server or the client and tell me if you wish to receive updates.

7.2 Notes

- The center of Gopher Space is `gopher.micro.umn.edu`, the FTP archive is `boombox.micro.umn.edu`.
- There exists a different implementation for VM, sometimes refered to as the 'Rice' VM Gopher System. RXSOCKET and PIPES are requiered for this one. For information about this system contact Rick Troth, his email address is `TROTH@RICEVM1.bitnet`.
- A Gopher server running under MVS also exists. (mvs.udel.edu) The author is Steve Bacher, his email address is `seb1525@mvs.draper.com`.
- The news group `comp.infosystems.gopher` (in former times `alt.gopher`) exists for discussion of Gopher in general, the Bitnet Listserv mailing list `VMGOPHER@PUCC.bitnet` was created for discussion of Gophers running on VM.
- Gopher has a couple of close or not so close relatives:
  - **CWIS**: Campuswide Information Systems
    ... some of them are Gophers in disguise, others would like to be.
  - **Gopher+**:
    Other Gophers think he's a snob, but he is not ...
    ... he's just a bit more sophisticated than the ordinary ones![9]
  - **PANDA**:
    ... a special breeding of Gophers that is now domestified in Iowa. The home of panda is `panda.uiowa.edu`. Some Gophers are in fact cross-breedings of the archaic Gopher generation and Pandas (e.g. VieGOPHER).

[9] details can be found in [2].

- **WAIS**: Wide Area Information Server
  ... a quite powerful information retrieval package that is often used as the search engine of choice in Gopher Space. WAIS-world has it's center in `quake.think.com` [10]. A very good introduction can be found in [8].
- **WWW**: World-Wide Web
  Internet-wide distributed hypertext system based on the HTTP protocol. (Hypertext transport protocol)

7.3 Acknowledgements

- The Gopher Team of the University of Minnesota
  (`gopher@micro.umn.edu`)
  The simple things prove always to be the best, great job!
- Ken Hornstein
  (`kxh105@psuvm.psu.edu`)
  Without his **REXTCPIP** neither the client nor the server would exist at all.
- Arthur Ecock (City University of New York, CUNY)
  (`eckcu@cunyvm.cuny.edu`)
  Let's hope more programs are designed, implemented and documented in such an excellent way like **RXSOCKET**.
- Alan J. Flavell (University of Glasgow)
  (`sy07@i1.ph.gla.ac.uk`)
  His suggestions were used to improve the code and to clean up some of the major irregularities.
- Mary Posey (University of Maryland, UMCP)
  (`mposey@umdacc.umd.edu`)
  She spent an enormous amount of time and energie to point out and remove inconsistencies in the code.
- John Sroka (San Jose State University)
  (`sroka@sjsuvm1.sjsu.edu`)
  Help? He wrote the help text, I just took it :)).
- ... and finally, this girl in Chicago for reminding me who the audience is.

[10] a lot of useful information about WAIS, e.g. [7] can be found there

8 Implementation Notes

This section gives informal notes about VieGOPHER, e.g. some variables and details of the servers and client modules. Neither completeness nor usefulness for any purpose can be assumed.

8.1 Discussion: Why are things the way they are?

8.1.1 Client: writing files to disk temporarily

I was asked more than once why the client writes files received from the server to disk first, reads it into editor memory and then discards the disk file. Alan Flavel suggested that `GOPHFTCH.EXEC` should push lines to the stack, wrapped into XEDIT red tape. At this point, finally, this shortcoming is more or less resolved.[11]
There were several reasons why the file write method was used:
First: the client started as a single module that had to fetch items and store them to disk. This was initially the only purpose of `GOPHFTCH.EXEC`. Later, as interactive XEDIT support was added, the implementation of `GOPHFTCH.EXEC` remained more or less intact.
Second: there are in fact items in Gopher Space that have record sizes of more than 256 bytes, e.g. directories from WAIS-to-Gopher gateways sometimes deliver records in the range of 600 bytes per line. A record on the stack can only be 255 byte long, so using the stack for holding incoming data items requires a method for breaking up long lines. This fact should explain the format of data on the stack described in figure 2 on page 12.

The current implementation of VieGOPHER features another method of bringing data to the screen. In a typical gopher session, the fetcher module `GOPHFTCH.EXEC` is called with the `XEDIT` file mode option. This option allows the fetcher to transfer data directly to XEDIT data buffer and commands XEDIT to refresh the screen after the first couple of lines in order to bring the data to screen as fast as possible. For long files this has the effect that a user can read the first screen while the rest of the file is still being transfered.

Unfortunately, this version still does not feature an interrupt method or a transparent interaction method, so the user is required to wait until the data transfer is finally over. During the transfer, XEDIT displays the number of lines transfered sofar in the message area.

For transfering index files that originate from a type 7 search, the write - read - erase method is used as described before. The reason for doing so originates from the fact that WAIS searches — these generate type 7 indexes — can produce enourmously long selector strings (paths). This compromise

[11] If you want more backgroud information about the way VieGOPHER operates in certain situations, please drop me a line...

allows to circumvent problems with extremely long lines which otherwise makes it impossibly to successfully process incoming data.

8.2 GOPHERDD.EXEC

The multi-threaded server handles several connections by using a number of so-called slots and a number of variables where information for each slot is recorded. All slots are checked and serviced periodically. The server starts by posting a LISTEN call for one or more different ports and assinging the status for LISTEN to these slots. When a connection on a port is established the server immediately assigns a new slot for that server type and post's the next listen call on the same port. Periodically, the program checks which slot it has to serve and eventually if a service is finished and the slot's status is changed to UNASSIGNED. If a connection crashes or behaves in some strange way the slot status is changed to dead and is not serviced any longer. A recovery strategy for the case of severe problems is currently in design.

8.2.1 variables describing information about a slot

`mtmax`	number of used slots
`mtact`	number of slot currently served
`mtgcn.`	connection number
`mtsta.`	slot status (see below)
`mtprt.`	port number assigned for the slot
`mttyp.`	type of the slot and consequently the service:
`mtfad.`	foreign address of the connection
`mtsel.`	selector string received for the slot
`mtsel1.`	prefix selector string for EXTRA slots
`mtsel2.`	postfix selector string for EXTRA slots
`mtmsg.`	message to be sent
`mtfil.`	file to be sent
`mtrec.`	records of the file sent sofar
`mtsiz.`	last number of record to be sent
`mtttw.`	maximum time to wait

8.2.2 slot types and ports

The following slot types define the service they are providing. Port numbers are officially assigned ones.

type	port	notes
GOPHER	70	the 'ordinary' Gopher server
FINGER	79	a Finger server
COOKIE	17	a Quote-of-the-Day server
SRE	150	(experimental)

8.2.3 status codes for a slot

0	`UNASSIGNED`	the slot is unused
1	`DEAD`	a connection crashed on that slot
1000	`LISTEN`	a LISTEN call is posted to the TCPIP system
1001	`LISTEN` again	after Trying to open resume with post-processing of Listen
1005	`Trying...`	after Listening a connection was coming up slowly, the status changed to `Trying to open`
1010	`GET-SELECTOR`	a selector string is expected on the now established connection
1020	`TRANSFER`	a file transfer is in progress
1090	`CLOSING`	the connection is in the process of closing
2000	`SRE LISTEN`	wait for a SRE connection

8.2.4 other variables

`deadcount`	number of dead slots

A Sample Installation

A.1 Virtual Machines, etc.

A.2 Profile of the virtual machine GOPHERD

```
/* ------------------------------------------------------------------ */
/* PROFILE EXEC for GOPHERD@AWIWUW11.bitnet                           */
/*                                                                    */
/* 1992 05 05                                                         */
/* ------------------------------------------------------------------ */
logfile='GOPHERD LOGFILE'
chief='GOPHER'

/* Act 1: do whatever might be useful */
'TERM HILIGHT ON'
'DTRIPF NOPAN'          /* who knows what this is good for? */
'TERM ESCAPE OFF'

/* Act 2: Log everything that happens */
'SPOOL CONS START'
'SPOOL CONS TO' chief

/* Act 3: Prepare the stage for the next show */
'SENDFILE' logfile chief
'ERASE' logfile
```

```
/* Act 4: link necessary disks */
parse value diagrc(8?,'LINK GOPHER 191 201 RR') with CPRc .
'ACCESS 201 B'
parse value diagrc(8?,'LINK TCPMAINT 592 197 RR') with CPRc .
'ACCESS 197 X'

/* Act 5: run the server now */
'GOPHERDD'              /* start the *MULTI THREADED* server now ... */

/* Act 6: finale? nobody knows .... */
```

A.3 GOPHERD.SELECTOR

```
* File GOPHERD SELECTOR
*
* list of known items
*
* 1992 03 04: <Gerhard.Gonter@wu-wien.ac.at>
* ---------------------------------------------------------------------
* root directory *****
ABOUT.GOPHER.B      gg * about
LOCAL.INDEX.B       gg * local
WUGOPHER.INDEX.B    gg * wu
ATGOPHER.INDEX.B    gg * austria
OTHER.INDEX.B       gg * others
EUROPE.INDEX.B      gg * EUROPE.INDEX
VMGOPHER.INDEX.B    gg * VM Gopher
DOCUMENT.INDEX.B    gg * documents
SERVICES.INDEX.B    gg * services
POINTERS.INDEX.B    gg * pointers
NETINFOO.INDEX.B    gg * net info
BSS.INDEX.B         gg * bss stuff
PROMARE.*.B         gg * PROMARE.INDEX
WAISES-1.INDEX.B    gg * WAISes (everything)
* VM Client & Server software directory *****
VMGOPHER.STATUS.B   gg * VMGOPHER.STATUS
VMGOPHER.DOC.B      gg * VMGOPHER.DOC
VMGOPHER.IMPLEMEN.B gg * VMGOPHER.IMPLEMEN
GOPHERD.EXEC.B      gg * GOPHERD.EXEC
GOPHERDD.EXEC.B     gg * GOPHERDD.EXEC
GOPHER.EXEC.B       gg * GOPHER.EXEC
GOPHER.XEDIT.B      gg * GOPHER.XEDIT
XG.EXEC.B           gg * XG.EXEC
GOPHFTCH.EXEC.B     gg * GOPHFTCH.EXEC
GOPHXFTC.XEDIT.B    gg * GOPHXFTC.XEDIT
GOPHXBMC.XEDIT.B    gg * GOPHXBMC.XEDIT
```

```
GOPHXBMI.XEDIT.B    gg * GOPHXBMI.XEDIT
GOPHXBMS.XEDIT.B    gg * GOPHXBMS.XEDIT
GOPHXSVI.XEDIT.B    gg * GOPHXSVI.XEDIT
GOPHXQIT.XEDIT.B    gg * GOPHXQIT.XEDIT
GOPHSRCH.XEDIT.B    gg * GOPHSRCH.XEDIT
EDINDEX.EXEC.B      gg * EDINDEX.EXEC
EDINDEX.XEDIT.B     gg * EDINDEX.XEDIT
GLOBALVV.EXEC.B     gg * GLOBALVV.EXEC
GLOBALXV.EXEC.B     gg * GLOBALXV.EXEC
FDNS.EXEC.B         gg * DNS.EXEC
FDNS.IN.B           gg * FDNS.IN
VMGOPHSF.EXEC.B     gg * VMGOPHSF.EXEC
* test index: WRITE PUBLIC ***************************
TEST.INDEX.B         * * test
***AUTOMATIC***
FORTUNE.TXT.B       gg * FORTUNE.TXT
```

A.4 GOPHERD.INDEX

```
0About␣Internet␣Gopher%about%awiwuw11.wu-wien.ac.at%70
1Local␣Info%local%awiwuw11.wu-wien.ac.at%70
1Gophers␣around␣the␣house%wu%awiwuw11.wu-wien.ac.at%70
1Gophers␣around␣the␣place%austria%awiwuw11.wu-wien.ac.at%70
1Gophers␣around␣the␣world%others%awiwuw11.wu-wien.ac.at%70
1documents%documents%awiwuw11.wu-wien.ac.at%70
1services%services%awiwuw11.wu-wien.ac.at%70
1external␣services%1/.exser.ind%olymp.wu-wien.ac.at%70
1life,␣the␣NET␣and␣everything%net␣info%awiwuw11.wu-wien.ac.at%70
1The␣Guide!%1/.guide.ind%olymp.wu-wien.ac.at%70
1VieGOPHER:␣VM/CMS␣Gopher␣Version%VM␣Gopher%awiwuw11.wu-wien.ac.at%70
1Test:␣use␣at␣your␣own␣risk%test%awiwuw11.wu-wien.ac.at%70
1Hypertext␣Documents␣␣(Items␣are␣Type=H)%1/.hyx%olymp.wu-wien.ac.at%70
```

Note: `%` represents the TAB character, hex code `'05'x` in EBCDIC (`0x09` in ASCII).

A.5 GOPHERD.GROUPS

```
* File GOPHERD GROUPS
*
* list of know groups
*
* 1992 03 10: <Gerhard.Gonter@wu-wien.ac.at>
* ----------------------------------------------------------------------
gg        137.208.
wu-wien   137.208.
```

```
edvz     137.208.16.
schnurli 137.208.16.39
```

A.6 GOPHERD.DSKGRP

```
* File GOPHERD DSKGRP
*
* list of known disk groups
*
* 1992 05 25: <Gerhard.Gonter@wu-wien.ac.at>
* ---------------------------------------------------------------------
B1      1       *
C1     10       umdacc schnurli
D2     12       wu-wien
E1      0       *
```

Comments:

- B1 is a public disk.
- C1 is available to groups `umdacc` and `schnurli` only.
- D2 is available to group `wu-wien` alone.
- E1 is unavailable because it has priority 0.

A.7 GOPHERD.EAI

```
* FILE GOPHERD EAI
*
* embedded application definition
*
* 1993-05-25
```

B Resource Guide

B.1 Packing List

The core of the VieGOPHER package consists of a number of modules that are grouped in four sub-packages. You should receive all modules in just one distribution file, arranged as a ROSE product file. The table below lists all modules, describes briefly what they do and what sub-package they belong to.

file name	SRV	CL	SUP	EXT	OBS	description
COOKIE.EXEC	X			X		produce a cookie message
EDINDEX.EXEC			X			edit Gopher index files
EDINDEX.XEDIT			X			mask for editing Gopher index
FINGER.EXEC				X		client interface for Finger
FINGERXX.EXEC	X			X		Finger response generator
GOPHCHRL.EXEC			X			check READERLIST for posted items
GOPHINST.EXEC			X			postinstaller calle by Rose
GOPHER.EXEC		X				client main module
GOPHER.XEDIT		X				setup for client main module
GOPHERDD.EXEC	X					server, multi-threaded
GOPHFTCH.EXEC		X				item fetcher
GOPHRING.XEDIT		X				present XEDIT ring list on screen
GOPHRNG3.XEDIT		X				display selected item from ring
GOPHSCAN.EXEC			X			scan a (group of) gopher server(s)
GOPHSCRN.XEDIT		X				screen splitting and joining
GOPHSRCH.XEDIT		X				mask for entering search string
GOPHSTOR.EXEC			X			store posted items
GOPHXBMC.XEDIT		X				save a bookmark (cursor)
GOPHXBMI.XEDIT		X				save a bookmark (displayed item)
GOPHXBMS.XEDIT		X				show bookmarks
GOPHXFTC.XEDIT		X				fetch selected item
GOPHXQIT.XEDIT		X				leave displayed item
GOPHXSVI.XEDIT		X				save displayed item
XG.EXEC		X				call XEDIT, Gopher style

B.2 Obtaining the Package

The VieGOPHER package can be obtained from a number of sources:

1. from the FTP server of the University of Minnesota (Gopher Central):
   `boombox.micro.umn.edu:/pub/gopher/Vienna_CMS/` or
   `boombox.micro.umn.edu:/pub/gopher/VieGOPHER/`

2. from the Gopher server of the University of Economics in Vienna:

```
Name=VieGOPHER: VM/CMS version
Path=VM Gopher
Type=1
Host=awiwuw11.wu-wien.ac.at
Port=70
```

3. from the FTP server of the University of Economics in Vienna:
   `ftp.wu-wien.ac.at:/pub/src/VieGOPHER/`

B.3 Obtaining Third-Party Modules

The following table gives a very brief overview over third-party modules, what they do and which sub-packages they are used for. Source files and documentation are not necessary for running the VieGOPHER system.

file name	SRV	CL	SUP	EXT	OBS	description
REXTCPIP.MODULE REXTCPIP.ASSEMBLE REXTCPIP.MEMO	X	X		X		TCP/IP transport package + source + description
RXSOCKET.MODULE REXXWAIT.MODULE		X				TCP/IP transport package general event handler
YWAKEUP.MODULE YWAKEUP.HELPCMS	X					trapping messages + description

Ken Hornstein's `REXTCPIP` can be retrieved from `LISTSERV@DEARN.bitnet` and a lot of other `LISTSERV`s around the globe.
Arthur Ecock's `RXSOCKET` and `REXXWAIT` are available from `LISTSERV@CUNYVM.bitnet`.
`YWAKEUP` is available from `LISTSERV@DEARN.bitnet` and other `LISTSERV`s.

C Sample Applications

C.1 Retrieving Information Automagically

This program retrieves information files containing weather reports and mails them to interested people.

```
/* retrieve weather reports and send them out */
parse arg usr

tmp='WEATHER TMP A'
srv='0 ashpool.micro.umn.edu 70'
ma1='0/Weather/Maine/Current Conditions'
ma2='0/Weather/Maine/Portland'
mas='0/Weather/Massachusetts/Current Conditions'
was='0/Weather/Washington/Seattle'
flo='0/Weather/Florida/Key West'
chi='0/Weather/Illinois/Chicago'
ill='0/Weather/Illinois'
ala='0/Weather/Alaska/Current Conditions'


'GOPHFTCH' tmp srv ma1
'PARCEL dm' tmp '(Maine weather'

'Sleep 10 sec'
'GOPHFTCH' tmp srv mas
'PARCEL gf' tmp '(Massachusetts weather'

'Sleep 10 sec'
'GOPHFTCH' tmp srv was
'PARCEL pp' tmp '(Washington weather'
```

```
'Sleep 10 sec'
'GOPHFTCH' tmp srv ala
'PARCEL pp' tmp '(Alaska weather'

'Sleep 10 sec'
'GOPHFTCH' tmp srv flo
'PARCEL gf' tmp '(Florida weather'

'Sleep 10 sec'
'GOPHFTCH' tmp srv chi
'PARCEL GONTER' tmp '(Chicago weather'

exit(0);
```

D List of files that need to be configured

D.1 ... by Rose

Most modules are configured at installation time by Rose. Most prominent among them are:

- `GOPHERDD.EXEC`
- `GOPHFTCH.EXEC`

Appendix E lists all Rose prompt texts that are used during that phase.

D.2 ... by hand (still)

- `FINGER.EXEC`
  The variable `localnode.1` contains the name of the local host.
- `FINGERXX.EXEC`
  The name of the phone book file is `PHONEBK.NAMES` user names are contained in `LOCAL.NAMES`.
- `COOKIE.EXEC`
  The fortune database consists of the files `FORTUNE.TXT` and `FORTUNE.IDX`.
- `TCPIP.DATA`
- `REXTCPIP.MODULE`

E Installation-time options

Server, client and other modules have options, that can be configured during installation time using ROSE. Here are the help texts and predefined values for these variables:

E.1 Server

```
#help MAKE_SERVER
MAKE_SERVER defines if server modules will be generated.
The preset value is YES.
#endhelp

#set MAKE_SERVER YES
# currently no prompting, because ROSE doesn't do it anyway...
# prompt MAKE_SERVER



#help LOGFILE
LOGFILE defines the file where server activities are logged.
The preset value can be overridden by a global variable of the same name
in the GOPHERD group.
#endhelp

#set LOGFILE 'GOPHERD LOGFILE A'
#prompt LOGFILE



#help SELECTORFILE
SELECTORFILE defines the file where all known selector strings are stored.
The preset value can be overridden by a global variable of the same name
in the GOPHERD group.
#endhelp

#set SELECTORFILE 'GOPHERD SELECTOR A'
#prompt SELECTORFILE



#help GROUPFILE
GROUPFILE defines the file where access group information is stored.
The preset value can be overridden by a global variable of the same name
```

```
in the GOPHERD group.
#endhelp

#set GROUPFILE 'GOPHERD GROUPS A'
#prompt GROUPFILE



#help EAIFILE
EAIFILE defines the file where handlers for embedded applications
are declard.
The preset value can be overridden by a global variable of the same
name in the GOPHERD group.
#endhelp

#set EAIFILE 'GOPHERD EAI A'
#prompt EAIFILE



#help DSKGRPFILE
DSKGRPFILE names the file where assignments between disks and
access groups are defined.
The preset value can be overridden by a global variable of the same
name in the GOPHERD group.
#endhelp

#set DSKGRPFILE 'GOPHERD DSKGRP A'
#prompt DSKGRPFILE



#help ROOTINDEX
ROOTINDEX defines the file that contains the root index or root menu.
This file is sent out whenever an empty selector string is presented
to the GOPHER server.
The preset value can be overridden by a global variable of the same name
in the GOPHERD group.
#endhelp

#set ROOTINDEX 'GOPHERD INDEX A'
#prompt ROOTINDEX
```

```
#help GOPHER_PORT
GOPHER_PORT is used in the server module to define the TCP port number
of the gopher server.  It is typically port number 70.
The preset value can be overridden by a global variable of the same name
in the GOPHERD group.
#endhelp

#set GOPHER_PORT 70
#prompt GOPHER_PORT



#help FINGER_PORT
FINGER_PORT is used in the server module to define the TCP port number
of the finger server.  It is typically port number 79.
The preset value can be overridden by a global variable of the same name
in the GOPHERD group.
#endhelp

#set FINGER_PORT 79
#prompt FINGER_PORT



#help COOKIE_PORT
COOKIE_PORT is used in the server module to define the TCP port number
of the cookie server.  It is typically port number 17.
The preset value can be overridden by a global variable of the same name
in the GOPHERD group.
#endhelp

#set COOKIE_PORT 17
#prompt COOKIE_PORT



#help USE_SRE
SRE means: Structured Representation Engine
This is an experimental protocol and usually you would not select it.
If you want to activate it, define the value YES, default is NO.
#endhelp

#set USE_SRE NO
#prompt USE_SRE
```

```
#help SRE_PORT
SRE_PORT is used in the server module to define the TCP port number
of the SRE server.  It is typically port number 150.
#endhelp
#set SRE_PORT 150
#prompt SRE_PORT



#help SRV_WAKEUP_SYSTEM
SRV_WAKEUP_SYSTEM defines if or if not messages (SMSG, MSG) from other
virtual machines or from RSCS are accepted and processed.
Typically, messages are trapped and processed.  In this case the
variable value is set to 'YWAKEUP' or to 'WAKEUP', depending on the
preference for either method.  Trapping messages with option YWAKEUP
requires YWAKEUP.MODULE, for WAKEUP you need IBM's WAKEUP module etc.
If no message trapping is wanted, enter NO.
If you have trouble using either method, select also NO.
#endhelp

#set SRV_WAKEUP_SYSTEM YWAKEUP
#prompt SRV_WAKEUP_SYSTEM



#help SRV_TCP
SRV_TCP defines the TCP transport package to be used.  Allowed value
is REXTCPIP (the original and default package).
RXSOCKET might be an option in the future.
#endhelp

#set SRV_TCP REXTCPIP
# prompt SRV_TCP
```

E.2 Client

```
#help MAKE_CLIENT
MAKE_CLIENT defines if client modules will be generated.
The preset value is YES.
#endhelp

#set MAKE_CLIENT YES
```

```
# currently no prompting, because ROSE doesn't do it anyway...
# prompt MAKE_CLIENT



#help STARTUP_DIR
STARTUP_DIR defines a string that is displayed as a title for the
first Gopher menu.

Note: This value is *not* entered a quoted string.

This setting can be re-defined by the user with the SETUP function
in Gopher or by calling GOPHSTP.
#endhelp

#set STARTUP_DIR WU-Wien Gopher Information Service: Main Menu
#prompt STARTUP_DIR



#help CL_TCP
CL_TCP defines the TCP transport package to be used by the client.
Allowed values are REXTCPIP (the original package) and
RXSOCKET (now the default).

Please report any problem that you encounter with
alternative TCP transport support.
#endhelp

#set CL_TCP RXSOCKET
#prompt CL_TCP



#help CL_USE_FDNS
CL_USE_FDNS defines if the FDNS hack should be used to resolve a gopher
hosts IP address or not.  Default is NO.
YES should be selected if you find it absolutely necessary to circumvent
the GetIPAddr() function in REXTCPIP.  If RXSOCKET is used, this variable
must be set to NO.
#endhelp

#set CL_USE_FDNS NO
# we don't allow this option any longer, it's not necessary ...
```

```
# prompt CL_USE_FDNS



#help CL_USE_LOCALFETCH
CL_USE_LOCALFETCH defines if the client should be allowed to access data
on the Gopher server's disk, if the server is on the very same host as
the client.

Localfetch is mainly a trade-off between data-privacy (whatever this
means when we speak about a server) and performance.  Using localfetch
you can also setup a pseudo-server, that is, you provide only gopher
data and don't run the server code.

YES     works if individual clients have read access to the gopher
        data files.  This is needed for pseude servers.
NO      should work in most cases.
Default is NO.
#endhelp

#set CL_USE_LOCALFETCH NO
#prompt CL_USE_LOCALFETCH



#help CL_LANG
CL_LANG defines the language that the client uses for messages to
the user and help screens.
Defined values are: ENGLISH and GERMAN, default is ENGLISH
If you want another language included in the client, contact the author.
#endhelp

#set CL_LANG ENGLISH
#prompt CL_LANG



#help CL_DEFAULT_HOST
CL_DEFAULT_HOST defines the Internet name of the default Gopher server
that should be contacted on startup.  Usually you will want to declare
one of your favorite servers next door but you can of course also use
the default value 'gopher.wu-wien.ac.at'.

Note:           Enter the host name as a quoted string!
Default:        'gopher.wu-wien.ac.at'
```

```
This setting can be re-defined by the user with the SETUP function
in Gopher or by calling GOPHSTP.
#endhelp

#set CL_DEFAULT_HOST 'gopher.wu-wien.ac.at'
#prompt CL_DEFAULT_HOST



#help CL_DEFAULT_PORT
CL_DEFAULT_PORT defines the TCP port number at which your Internet
Gopher server listens.  The 'official' port number is now 70 and
so is the default value.

Note:           the port number must be entered as a quoted string.
Default:        '70'

This setting can be re-defined by the user with the SETUP function
in Gopher or by calling GOPHSTP.
#endhelp

#set CL_DEFAULT_PORT '70'
#prompt CL_DEFAULT_PORT



#help CL_DEFAULT_TYPE
CL_DEFAULT_TYPE defines the code for the item type that you'll receive
from your Gopher server.  According to the protocol, a Gopher server
should return a directory, if an empty selector string is sent to the
server.  The code for directory is '1' and so is the default value for
this variable.

Note:           Values other than '1' are not recommended.
Default:        '1'

This setting can be re-defined by the user with the SETUP function
in Gopher or by calling GOPHSTP.
#endhelp

#set CL_DEFAULT_TYPE '1'
#prompt CL_DEFAULT_TYPE
```

```
#help CL_TIME_OUT
CL_TIME_OUT defines the time in seconds that the client will wait
for the server to respond.  After this timeout, the client will
simply display a message about the unavailable server.
Default value is 15.
#endhelp

#set CL_TIME_OUT 15
#prompt CL_TIME_OUT



#help CL_INFO_LEVEL
CL_INFO_LEVEL defines the default level of information that is in
effect when the gopher client displays menus or files etc.

Currently defined values are:
0 ... no function keys
5 ... two lines of function key description     (DEFAULT)
9 ... four lines of function key description

This setting can be re-defined by the user with the SETUP function
in Gopher or by calling GOPHSTP.
#endhelp

#set CL_INFO_LEVEL 5
#prompt CL_INFO_LEVEL



#help CL_CONTACT
CL_CONTACT defines the email address of a person that might be able to
answer to typical questions of users.
The default value 'Gerhard.Gonter@wu-wien.ac.at'.
#endhelp

#set CL_CONTACT Gerhard.Gonter@wu-wien.ac.at or GONTER@AWIWUW11.bitnet
#prompt CL_CONTACT



#help CL_TELNET_8
CL_TELNET_8 defines the name of the program for use with Telnet session
pointed to by gopher item type 8.  Basically, the gopher protocol does not
```

```
specify, which terminal emulation should be used and it would be a nice
feature if the Telnet program could automatically switch to VT100 or
3270 mode as appropriate.  Since this is not done by IBM's Telnet you
have to trade off between the various disadvantages.

A possible choice for CL_TELNET_8 is 'TNVT100 EXEC', Arthur Ecock's
VT100 emulater.  This program requires TCPIP Version 2 and it might
conflict with REXTCPIP in some (most?) installations.

Note:           Enter the name of the program as a quoted string
                and specify the filename *AND* the filetype
Default:        'TELNET MODULE'

This setting can be re-defined by the user with the SETUP function
in Gopher or by calling GOPHSTP.
#endhelp

#set CL_TELNET_8 'TELNET MODULE'
#prompt CL_TELNET_8



#help CL_TELNET_T
CL_TELNET_T defines the name of the program for use with Telnet session
pointed to by gopher item type T.  Basically, the gopher protocol does not
specify, which terminal emulation should be used and it would be a nice
feature if the Telnet program could automatically switch to VT100 or
3270 mode as appropriate.  Since this is not done by IBM's Telnet you
have to trade off between the various disadvantages.

Type T is now an official extension to the original Gopher type list and
is mostly used for 3270 based telnet sessions.  Thus, in most situations
the default value 'TELNET MODULE' would be ok.  But you have the choice...

Note:           Enter the name of the program as a quoted string
                and specify the filename *AND* the filetype
Default:        'TELNET MODULE'

This setting can be re-defined by the user with the SETUP function
in Gopher or by calling GOPHSTP.
#endhelp

#set CL_TELNET_T 'TELNET MODULE'
#prompt CL_TELNET_T
```

F Question and Answers

F.1 How does the server check access groups?

Question

> How would the server know that the maryp.listing.e would be in the umcp group and mary1.listing.e would be in the * group?

Answer

Here's how the authentication (great word, isn't it?) algorithm works:

1. the client (software) sends the selector string.
2. the server tries to find a matching entry in the selector file, possibly with wildcards for filenames.
3. if the server finds a matching entry it checks the group:
   (a) if the data item is for group `*`, it is sent over the line.
   (b) if there is a group info included in the selector record (in the file `GOPHERD.SELECTOR`), the server checks if the client (computer) belongs to the same group. The **only** way this can be achived at all is by comparing the IP address of the client computer. Basically, the same method is used in the unix server too.
   (c) if the group entry does not match, the server goes back to step 2. and examines the next records from the selector file.
4. If no matching entry is found in the selector file, the server returns a generic error message, indicating that no entry was found.

Please note:

- It's the server's responsibility to check the propper access rights. Never trust your clients, they often turn out to be your enemies...

F.2 How does the disk group system work?

Question

> How does the disk group system work, and how does the server know if it can send a file to a privileged user or not?

Answer Say you have a menu where a selector string `FILE-XYZ.ABC` is defined, and you have the file `XYZ.ABC` on these disks:

```
Filename Filetype Fm ... Lrecl ... Date ... etc
XYZ      ABC      C2
XYZ      ABC      D6
```

Say you have these entries in the disk groups file `GOPHERD.DSKGRP`:

```
C2    1   *
D1    2   *
D6   10   umdacc
```

and the groups file GOPHERD.GROUPS says:

```
umdacc 199.99.
```

or so, whatever is correct for you.

Let's assume, an user from `122.22.22.2` sends the string `FILE-XYZ.ABC`, what does the server? He looks up for all files `XYZ ABC` on all disks, sees that filemode `D6` is available, checks it in the disk groups file, sees that the clients IP address does **NOT** match the group id in the groups file. Thus this file is not used. It further sees that the file is also on `C2`, so it checks if this group, * is public so it would deliver the file `XYZ ABC C2`.

Let's now do an user from `199.99.99.9` do the same thing. The server would see that `C2` would be possible, with a priority of 1, but there is also `D6` with a priority of 10. The server also sees that `D6` should be visible to `199.999.` so it would accept this file and since `D6` has a higher priority than `C2` it would ship the file `XYZ ABC D6` to the umdacc-user.

F.3 How can a pseudo server be set up?

Question

What is a pseudo server good for and how can it be set up?

Answer The purpose of a pseudo server is to provide Gopher menus without actually running a real Gopher server. This will be of interest for installations, where no real campus server is available and a VieGopher server will not be installed, for whatever reason.

A system administrator can, for instance, set up a short menu where users can select between a few local information files and entrypoints to various other (real) servers.

Setting up a pseudo server should be fairly easy:

1. When installing the system with `ROSE VIEGOPHR` answer the question in the `CL_USE_LOCALFETCH` prompt `YES`. If you installed the product already and answered the question with `NO` (as is the default) you can remove the line
   `FIX CL_USE_LOCALFETCH NO`
   from the file `VIEGOPHR FIX` and run Rose again.

2. ROSE also creates a number of sample files and renames them to the default value, if they are not present already. These files can then be modified to reflect the local setup.

generated	renamed	purpose
`GOPHERD.SINDEX`	`GOPHERD.INDEX`	Root menu for the pseudo server
`GOPHERD.SSELECT`	`GOPHERD.SELECTOR`	assigns known path strings with real file names
`GOPHERD.SABOUT`	`GOPHERD.ABOUT`	short description of Gopher, displayed as file

G Author's Address

I can be reached — for questions, remarks, job offers, etc. — in the following ways:

email: <Gerhard.Gonter@wu-wien.ac.at>
<GONTER@AWIWUW11.BITNET>

snail mail: Gerhard Gonter
University of Economics, Vienna
Augasse 2-6
A-1090 Wien
Austria, Europe

fax: +43/1/31336/702

phone: +43/1/31336/4111

H Internal To-Do List

H.1 Installation

- REXTCPIP.MODULE, RXSOCKET.MODULE, YWAKEUP.MODULE on awiwuw11 for ftp
- $\boxed{\text{TEST}}$ postinstaller: prompting for immediate ordering of the modules from the servers

H.2 Client

- HELP on PF1
  adaption of John Sroka's help text.
- ROSE setup for unasigned function keys
- ROSE setup for Save and Print Keys
- interface for a print function
- setup screen for user option selection
  - download options
  - global variables for server, etc...
- Retrieval/Transfer Options:
  - destination file
  - retrieve text or binary to file
  - blocking and record length
  - transparent BSS, UU, XX or HQX retrieval
  - special options, e.g. for DVI files or some other well known file formats
- Item Inspector and Direct Access:
  Show/Edit information about an item, editor mask for
  - Name
  - Host
  - Port
  - Path
  - Type

  This could possibly be implemented as a derivation of **EDINDEX.XEDIT** or **EDINDEX.EXEC**.
  It might be also useful in a bookmark editor environment.
- in bookmark list, function key for deleting entries

H.3 `GOPHFTCH.EXEC`

- (*) a converter for dotted IP-addresses is needed.
- interrupting data transfer and/or parallel retrieval
- transfer limitation with user setup:
  - number of lines
  - number of seconds for the transfer
  - combination of line count and transfer time expressed in some formula

  defaults: no limits
- no NUCXDROP during a gopher session

H.4 Server

- improved stabilization for extremely agressive clients like the Macintosh Turbo Gopher. The problem actually is located somewhere in the REXTCPIP code, this is a 4500 lines assembler file...
- converting FINGER and COOKIE calls to EAI
- $\boxed{\text{TEST}}$ Diskgroups: reading, usage, etc...

H.5 EDINDEX

- export function; write all records to a file in the unix format. (Name, Path, Type, Host, Port)

H.6 documentation

- bookmarks:
  - PF3 saving the bookmark file
  - editing the bookmark file
  - structure of the save file
- client functions and keys:
  - screen split
  - history
  - root menu
  - Gnu General Public Licenese??
  - complete cross references to calling gopher(*) and gophftch

H.7 `FINGERXX`

Using of **`NAMEFIND`**:
in **`chknames()`**:

type	lookup by	returned
	:phone	:cnt :oid :nick
first name	:fname	:cnt :oid :nick
full name	:name	:cnt :oid :nick
	:nick	→ fields

in **`show_usr()`**: lookup by :terminal.

H.8 External Modules

- timing problems in `REXTCPIP.MODULE`
- error and exception handling in `REXTCPIP.MODULE`
- alternative support for `RXSOCKET.MODULE` instead of `REXTCPIP.MODULE`

H.9 checklist

- binary transfer for types I and 9, implemented already????
- WAKEUP-function in GOPHERDD

H.10 using RXSOCKET for the server

- emulating of connstat and conn_stat
- non-blocking listen: ok!
- non-blocking accept: not ok!
- checkout select and/or REXXWAIT

References

[1] B. Alberti, F. Anklesaria, P. Lindner, M. McCahill, D. Torrey. *The internet Gopher protocol — a distributed document search and retrieval protocol.* University of Minnesota. FTP'ed from boombox.micro.umn.edu. Spring 1992.

[2] B. Alberti, F. Anklesaria, P. Lindner, M. McCahill, D. Torrey. *Gopher+ — proposed enhancements to the internet Gopher protocol.* University of Minnesota. FTP'ed from boombox.micro.umn.edu. Summer 1992.

[3] F. Anklesaria, M. McCahill, P. Lindner, D. Johnson, D. Torrey, B. Alberti. *RFC1436: The Internet Gopher Protocol (a distributed document search and retrieval protocol)*. Network Working Group. March 1993.

[4] G. Gonter. *VieGOPHER — A Gopher System for VM/CMS*. Wirtschaftsuniversität Wien. 1992.

[5] K. Hornstein. `REXTCPIP.ASSEMBLE` *Version 1.8.4*. Pennsylvania State University. 1991.

[6] K. Hornstein. `REXTCPIP.MEMO` *Using the Rexx TCP/IP interface. (Version 1.8)*. Pennsylvania State University. 1991.

[7] B. Kahle. *Wide Area Information Server Concepts*. Thinking Machines Corp. FTP'ed from `quake.think.com:/pub/wais`. November 1989.

[8] P. Marshall. *WAIS: The Wide Areo Information Server — or — Anonymous What???* University of Western Ontario. FTP'ed from `ftp.uwo.ca`. June 1992.

[9] J. Postel. *RFC865: Quote of the Day Protocol*. Network Working Group. May 1983.

[10] D. Zimmerman. *RFC1288: The Finger User Information Protocol*. Network Working Group. December 1991.