VieGOPHER

—

A Gopher System for VM/CMS

—

The Code

Gerhard Gonter

June 16, 1993

Abstract

The Internet Gopher Protocol provides a simple, yet very effective method for distributed document search and retrieval. This document contains a documented source listing of the ViеGOPHER system. Included are the sources of the server, the client and the ViеGOPHER specific support programs and files. The printed version was generated from a Rose product file (`VIEGOPHR.PRODUCT`). The product file itself was generated by linearizing a portion of a NED hypertext file that contains the documentation, the code and all extra information. The hypertext editor NED as well as the hypertext file for the ViеGOPHER system can be obtained from `Gerhard.Gonter@wu-wien.ac.at`.

Contents

1 Introduction

```
ROSE file: viegophr.product

This file contains most files related to the VieGOPHER system.
To install the product enter the command: ROSE VIEGOPHR

% NOTE: This is not the original version of the document!
%       Instead, this is the sequenced and stripped version of a
%       hypertext file.
%       The linear ROSE version is provided for your convenience.
%       For more information contact the author.


#set       RELEASE VieGOPHER Ver 2.00.00
#set  RELEASE_DATE 1993-06-16
#set  RELEASE_TIME 18:00
\def\LPbotC{VieGOPHER 2.00.00 1993-06-16 18:00}

#set SENDFIX NO
```

2 Help messages

2.1 *help* TEST_and_HACK

```
#help TEST_and_HACK
There might be a number of Test and Hack options that can be configured.
Usually you should select the predefined value.  No further details about
any option is given.
#endhelp

#set TEST_and_HACK YES
#prompt TEST_and_HACK
```

2.2 *help* TEST_dd_binsel

```
#help TEST_dd_binsel
TEST_dd_binsel:
BIN ... REXTCPIP, selector string is read binary and converted
#endhelp
#set TEST_dd_binsel NO
#prompt TEST_dd_binsel
```

2.3 *help* Compactor

```
#help Compactor
Compactor allows the installation procedure to remove unrelevant parts
from the EXEC and XEDIT files.  Removed are comments, extra spaces,
and lines.  This reduces loading time a bit and might bring a little
bit of performance.  Compacting will only be executed if the program
REXXCPCX.MODULE is present.

Possible options are:
NONE ... to compact nothing at all (the default)
ALL  ... to compact everything
C    ... to compact just client modules
S    ... to compact server modules
```

```
T    ... to compact tools
Answering CU allows the compacting of the client and the utilities
alone, the server will not be modified.  CST is equivalent to ALL.
#endhelp

#set Compactor NONE
#prompt Compactor
```

2.4 *help* MAKE_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


# This is a block of definitions for the server code.
# switch MAKE_SERVER
# case YES
```

2.5 *help* LOGFILE

```
#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
```

2.6 *help* SELECTORFILE

```
#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
```

2.7 *help* GROUPFILE

```
#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
```

2.8 *help* EAIFILE

```
#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
```

2.9 *help* DSKGRPFILE

```
#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
```

2.10 *help* ROOTINDEX

```
#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
```

2.11 *help* GOPHER_PORT

```
#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
```

2.12 *help* FINGER_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
```

2.13 *help* COOKIE_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
```

2.14 *help* USE_SRE

```
#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
```

2.15 *help* SRE_PORT

```
#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
```

2.16 *help* SRV_WAKEUP_SYSTEM

```
#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
```

2.17 *help* SRV_TCP

```
#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
```

2.18 *help* SRV_ORGANIZATION

```
#help SRV_ORGANIZATION
SRV_ORGANIZATION is the short name of your organization.
Don't use single quotes in the string.
#endhelp

#set SRV_ORGANIZATION Univ. of Economics, Vienna, Austria
#prompt SRV_ORGANIZATION


# endswitch
# this ends the block of fixes for the server modules
```

2.19 *help* MAKE_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



# This is a block of definitions for the client code.
# switch MAKE_CLIENT
# case YES
```

2.20 *help* STARTUP_DIR

```
#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
```

2.21 *help* CL_TCP

```
#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
```

2.22 *help* CL_USE_FDNS

```
#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
```

2.23 *help* CL_USE_LOCALFETCH

```
#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
```

2.24 *help* CL_LOCALHOST

```
#help CL_LOCALHOST
CL_LOCALHOST containts the IP address of the local machine.  This
information is only relevant when the option CL_USE_LOCALFETCH is
answered with YES.  In this case the client will try to read the
gopher servers files direclty from his minidisk(s) without
establishing a real TCP connection to the server.

As fix value enter your machine's IP address.  If you don't intend
to allow direct access to the servers data you will not need to
give the real IP address.

Minor bug: if you're using RXSOCKET you should enter your hostname
instead of the IP address
#endhelp
```

```
#set CL_LOCALHOST 999.99.9.99
#prompt CL_LOCALHOST
```

2.25 *help* CL_LANG

```
#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
```

2.26 *help* CL_DEFAULT_HOST

```
#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
```

2.27 *help* CL_DEFAULT_PORT

```
#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
```

2.28 *help* CL_DEFAULT_TYPE

```
#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
```

2.29 *help* CL_TIME_OUT

```
#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
```

2.30 *help* CL_INFO_LEVEL

```
#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
```

2.31 *help* CL_CONTACT

```
#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
```

2.32 *help* CL_TELNET_8

```
#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
```

2.33 *help* CL_TELNET_T

```
#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


# endswitch
# this ends the block of fixes for the client modules


# This is a block of definitions for the server code.
# switch MAKE_SERVER
# case YES
```

3 Multi-Threaded Server

3.1 Module GOPHERDD EXEC

```
#erase GOPHERDD EXEC
#module GOPHERDD EXEC
```

3.1.1 main init

```
/* ------------------------------------------------------------------ */
REVISION='GOPHERDD EXEC #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME>#'
/*                                                                    */
/* an experimental multi-threaded multi-protocol daemon for VM/CMS:   */
/* 1. GOPHER         (protocol description spring 1992)               */
/* 2. FINGER         (RFC1288 compliant)                              */
/* 3. COOKIE         (RFC865 compliant Quote-of-the-Day)              */
#switch USE_SRE
#case YES
/* 4. S-REP ENGINE    (experimental)                                  */
#endswitch
/*                                                                    */
/* Uses:                                                              */
/*   REXTCPIP MODULE                                                  */
#switch SRV_WAKEUP_SYSTEM
#case YWAKEUP
/*   YWAKEUP  MODULE                                                  */
#case WAKEUP
/*   WAKEUP   MODULE                                                  */
#endswitch
/*   FINGERXX EXEC                                                    */
/*   COOKIE   EXEC                                                    */
/*   REACCESS EXEC                                                    */
/*                                                                    */
/* written:       1992-05-03: <Gerhard.Gonter@wu-wien.ac.at>          */
/*                1992-06-13: <> SRE: Structured Representation Engine */
/*                1993-04-14: <> EAI: Embedded Application Interface  */
/* latest update: 1993-05-27                                          */
/* ------------------------------------------------------------------ */

'GLOBALV SELECT GOPHERD STACK LOGFILE SELECTORFILE GROUPFILE EAIFILE',
  'DSKGRPFILE GOPHERPORT FINGERPORT COOKIEPORT SREPORT ROOTINDEX'
pull logfile
pull selectorfile
pull groupfile
pull eaifile
pull dskgrpfile
pull gopherport
pull fingerport
pull cookieport
pull sreport
pull rootindex

/* to be configured at installation time by ROSE:  - - - - - - - - - - */
if logfile=''      then logfile=      #<LOGFILE>#;
if selectorfile='' then selectorfile= #<SELECTORFILE>#;
if groupfile=''    then groupfile=    #<GROUPFILE>#;
if eaifile=''      then eaifile=      #<EAIFILE>#;
if dskgrpfile=''   then dskgrpfile=   #<DSKGRPFILE>#;
```

```
if rootindex=''   then rootindex=   #<ROOTINDEX>#;
if gopherport=''  then gopherport=  #<GOPHER_PORT>#;
if fingerport=''  then fingerport=  #<FINGER_PORT>#;
if cookieport=''  then cookieport=  #<COOKIE_PORT>#;
#switch USE_SRE
#case YES
if sreport=''     then sreport=     #<SRE_PORT>#;
#endswitch
/* - - - - - - - - - - - - - - - - - - -  end of configuration section */

stopflag=0;             /* 1 -> stop execution ...                    */
rwdisk='A';             /* minidisk for R/W access                    */

selcnt=0;               /* number of known GOPHER selectors           */
selid.=''               /* list of known selectors                    */
selfil.='';             /* associated file for the selector           */
selgrp.='';             /* group which can access the given file      */

grpcnt=0;               /* number of known gropus                     */
grpid.='';              /* access group name                          */
grpaddr.='';            /* IP addresses that are members of a group   */

dskgrpcnt= 0;           /* number of known disk groups                */
dskgrpnam.= '';         /* group name for this disk group             */
dskgrppri.= 0;          /* priority for this disk group               */
dskgrpdsk.= '';         /* disk mode for this group                   */

TTW= 600;               /* time to wait (10 minutes)                  */
MAXSQ= 16384;           /* maximum output queue size                  */
recsblk= 64;            /* records sent in one block                  */
dosleep= 0;             /* to sleep or not to sleep? ...              */
gsh_wait_exists= 0;     /* Gopher Server Hook for WAIT                */
gsh_wait_delay= 600;    /* number of idle cycles before executing     */
                        /* Gopher Server Wait hook code               */
gsh_wait_ticks= 0;      /* wait ticks elapsed sofar                   */

CRLF= '0D'x;
TAB= '05'x;

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
#switch SRV_WAKEUP_SYSTEM
#case YES
----> this mode is invalid, select either YWAKEUP or WAKEUP
#case YWAKEUP
'set msg off'
'set smsg off'
'set wng off'

'YWAKEUP BEGIN *MSG SMSG'
'set  msg iucv'
'set smsg iucv'
#case WAKEUP
'set msg off'
'set smsg off'
'set wng off'

'WAKEUP +0 ( IUCVMSG'
'set  msg iucv'
'set smsg iucv'
```

```
#endswitch

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
nucx_loaded= 0;
RECOVER:
mtmax= 0;               /* number of initialized servers              */
if nucx_loaded/=0 then 'NUCXDROP REXTCPIP';
x= logtext('***** starting multi-threaded GOPHERDD daemon *****');
'REXTCPIP'                            /* initialize REXTCPIP MODULE    */
nucx_loaded= 0;

x= reaccess();
# x= readselectors(selectorfile);       /* initialize selector list      */
# x= readgroups(groupfile);             /* initialize group list         */
# x= read_eai(eaifile);                 /* initialize EAI handler list   */

should_listen=,                         /* slightly modified Gopher      */
  startserver('GOPHER', gopherport, '', '');
should_listen= should_listen+,          /* almost normal Finger server   */
  startserver('FINGER', fingerport, '', '');
should_listen= should_listen+,          /* simple Cookie server          */
  startserver('COOKIE', cookieport, '', '');
#switch USE_SRE
#case YES
should_listen= should_listen+,          /* exper. S-REP Engine           */
  startserver('SRE',    sreport,    '', '');
#endswitch
```

3.1.2 main loop

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
do forever;          /* GOPHER ... a neverending story (hopefully) ... */

  if dosleep=1 then do;
    'SLEEP 1 SEC'
    /* Gopher Server Wait hook code */
    gsh_wait_ticks= gsh_wait_ticks+1;
    if gsh_wait_exists=1 & gsh_wait_ticks=gsh_wait_delay then do;
say date() time() 'GSH_WAIT';
      'GSH_WAIT';
      gsh_wait_ticks= 0;
    end;
  end;
  dosleep=1;
  deadcount=0;
  emptycount=0;
  listencount=0;

  do mtact=1 to mtmax;
    sta=mtsta.mtact;
    gcn=mtgcn.mtact;

    select; /* scheduler :)) */
      when sta=0 then do; /* unassigned slot */
        emptycount=emptycount+1;
      end;
      when sta=1 then do; /* dead slot; recycle */
        deadcount=deadcount+1;
        mtsta.mtact=0;
```

```
      say 'recycling dead slot['mtact']'
    end;

    /* listen as gopher, finger or cookie srv */
    when sta=1000|sta=1001 then do;
      CONNSTAT(gcn);
      if CONN_STAT='Listening' then do;
        listencount=listencount+1; /* nothing else ... */
      end; else do;
        if sta=1000 then do;
          /* fork() CMS style */
  x= startserver(mttyp.mtact, mtprt.mtact, mtsel1.mtact, mtsel2.mtact);
          x= logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
               'status changed from listening to' CONN_STAT);
          listencount=listencount+1;
        end;
        select;
          when CONN_STAT='Connected' then do;  /* we have contact! */
            x= logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                 'connected to' foreign_addr':'foreign_port);
            if mttyp.mtact='COOKIE' then do;
              x= gopherspecial(mtact 'SPEC-COOKIE');
              mtsta.mtact= 1020;
            end; else mtsta.mtact= 1010;
            mtttw.mtact= TTW;
            mtfad.mtact= foreign_addr;
            dosleep= 0;
          end;
          when CONN_STAT='Closed' |,
               CONN_STAT='Non-existant' then do;
            x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                      '('sta') unexpected connstat:' CONN_STAT);
            mtsta.mtact= 0;
          end;
          when CONN_STAT='Trying to open' then do;
            say 'Hey, someone''s trying to open!'  /* nothing */
            mtsta.mtact= 1005;
            mtttw.mtact= TTW;
          end;
          when CONN_STAT='Receiving only' |,
               CONN_STAT='Sending only' then do;
            x= logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                       '('sta') unexpected connstat:',
                       CONN_STAT '-> now dead');
            mtsta.mtact=1;
          end;
          otherwise do;
            x= logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                       '('sta') unknown connstat:',
                       CONN_STAT '-> now dead');
            mtsta.mtact=1; /* now officially declared dead */
          end;
        end/*select*/;
      end/*else do*/;
    end/*when sta=1000*/;

    when sta=1005 then do;    /* after listening, the slot changed   */
      CONNSTAT(gcn);          /* to trying to open...                */
      select;
```

```
          when CONN_STAT='Trying to open' then do;
            /* wait a \LaTeX\ \LaTeX\ little bit longer ... */
            say 'sta=1005, still trying to open...'
            mtttw.mtact=mtttw.mtact-1;
            if mtttw.mtact=0 then mtsta.mtact= 1;
          end;
          when CONN_STAT='Listening' then mtsta.mtact= 1;
          otherwise mtsta.mtact= 1001;
        end/*select*/;
      end;

      when sta=1010 then do; /* waiting for a selector string */
        connstat(gcn);
        select;
          when CONN_STAT='Listening' then do;
            x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                      '('sta') unexpectedly listening!');
            mtsta.mtact=1000;
          end;
          when CONN_STAT='Connected' then do;
            mtttw.mtact=mtttw.mtact-1;
            yy=0;
say '1010: bytes_to_read='bytes_to_read 'ttw='mtttw.mtact;
            if bytes_to_read>0 then do;
              mtttw.mtact=mtttw.mtact+TTW;
#switch TEST_dd_binsel
#case BIN
              tcpreceive(gcn,bin); /* ## BINARY ## */
#default
              tcpreceive(gcn);
#endswitch
tl0=TCPLINE.0;
say 'gcn='gcn'   TCPRC='TCPRC'   TCPLINE.0='TCPLINE.0   'tl0='tl0
say 'length(tcpline.1)='length(tcpline.1)
              if TCPRC      = 0           &,
                 tl0       /= 'TCPLINE.0' &,
                 tcpline.0 >= 1           then do;
                /* strange error (REXTCPIP): if we get a line with the */
                /* length of exactly 8228 byte we really only get JUNK */
                /* cure: a) either deliver a message and tell the user */
                /* to try it again, b) deliver the root menu ...       */
                /* I choose a) for now...                              */
                if length(tcpline.1)>3000 then do;
                  yy= 3;
                end; else do;
#switch TEST_dd_binsel
#case BIN
                  sel=A2E(tcpline.1); /* ## BINARY ## */
                  parse var sel sel '0D'x .
say 'sel='sel
                  sel=strip(sel);
#default
                  sel=strip(tcpline.1);
#endswitch
                  say 'bytes_to_read='bytes_to_read 'sel='sel
                  if TCPRC=0 then mtsel.mtact=sel;
                  yy= gopherselector(mtact sel);
                  dosleep= 0;
                end;
```

```
          end;
        end;
        if mtttw.mtact=0 then yy= 99;
        select;
          when yy= 0 then do; /* just wait ... */ end;
          when yy= 1 then do;
            TCPSEND(gcn, mtmsg.mtact);
            mtsta.mtact= 1090;
          end;
          when yy=2 then mtsta.mtact= 1020;
          when yy=3 then do;
            x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                      '('sta') trouble 1010!');
            mtsta.mtact= 1090;
          end;
          otherwise do;
            x= logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                       '('sta') crash, waited too long!');
            mtsta.mtact= 1;
          end;
        end;
      end;
      when CONN_STAT='Closed' |,
           CONN_STAT='Non-existant' then do;
        x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                  '('sta') unexpected connstat:' CONN_STAT);
        mtsta.mtact= 0;
      end;
      when CONN_STAT='Receiving only' |,
           CONN_STAT='Sending only' then do;
        x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                  '('sta') unexpected connstat:',
                  CONN_STAT '-> now dead');
        mtsta.mtact= 1;
      end;
      when CONN_STAT='Trying to open' then do;
        /* nothing */
        say 'sta=1010, trying to open???? very strange...'
      end;
      otherwise do;
        x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                  '('sta') unknown connstat:',
                  CONN_STAT '-> now dead');
        mtsta.mtact=1; /* now officially declared dead */
      end;
    end/*select*/
  end/*when sta=1010*/;

  when sta=1020 then do; /* in the process of sending a file */
    CONNSTAT(gcn);
    select;
      when CONN_STAT='Listening' then do;
        x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                  '('sta') unexpectedly listening!');
        mtsta.mtact=1000;
      end;
      when CONN_STAT='Connected' then do;
        if unacked_bytes < MAXSQ then do;
          rec= mtrec.mtact;
```

```
            recs= recsblk;
            if rec+recs >= mtsiz.mtact then do;
              recs= mtsiz.mtact-rec;
              mtsta.mtact= 1090;  /* the last block of the file */
/* say '['mtact']' mttyp.mtact'('mtprt.mtact')' sta '-> 1090'; */
            end;
            'EXECIO' recs 'DISKR' mtfil.mtact rec+1 '(FINIS STEM LIN.'
            if LIN.0 < recs then do; /* problems? */
              recs= LIN.0;
              mtsta.mtact= 1090;  /* the last block of the file */
            end;
            do i=1 to recs;
              line= LIN.i;
              if substr(line,1,1)='.' then line='.'line
              tcpsend(gcn,line);
            end;
            mtrec.mtact= rec+recs;
            dosleep= 0;
          end; /* else say '('mtact')unacked_bytes='unacked_bytes */
        end;
        when CONN_STAT='Closed' |,
             CONN_STAT='Non-existant' then do;
          x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                    '('sta') unexpected connstat:' CONN_STAT);
          mtsta.mtact= 0;
        end;
        when CONN_STAT='Receiving only' |,
             CONN_STAT='Trying to open' then do;
          x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                    '('sta') unexpected connstat:',
                    CONN_STAT '-> now dead');
          mtsta.mtact=1;
        end;
        when CONN_STAT='Sending only' then do;
          x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                    '('sta') connection ended by other side');
/* say '*** ('mtact') status: 1020->1090' */
          mtsta.mtact=1090;
          dosleep=0;
        end;
        otherwise do;
          x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                    '('sta') unknown connstat:',
                    CONN_STAT '-> now dead');
          mtsta.mtact=1; /* now officially declared dead */
        end;
      end/*select*/
    end/*when sta=1020*/;

    when sta=1090 then do; /* trying to close the connection */
      CONNSTAT(gcn);
x=xconnstat(gcn 'Status 1090');
      select;
        when CONN_STAT='Listening' then do;
          x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                    '('sta') unexpectedly listening!');
          mtsta.mtact=1000;
        end;
        when CONN_STAT='Connected' |,
```

```
              CONN_STAT='Sending only' then do;
           i=100;
           if mttyp.mtact='GOPHER' then tcpsend(gcn,'.');
           do while (CONN_STAT='Connected' & bytes_to_read > 0);
             tcpreceive(gcn);
say 'TCPRC='TCPRC 'TCPLINE.0='TCPLine.0
             if tcpline.0 <= 0 then leave;
say 'TRASH:' tcpline.1
             i=i-1;
             if i<= 0 then leave;
             CONNSTAT(gcn);
           end;
           tcpclose(gcn);    /* now close the tcp connection */
           x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact') closing');
           mtsta.mtact=0;
           dosleep=0;
'CP Q TIME'
         end;
         when CONN_STAT='Closed' |,
              CONN_STAT='Non-existant' then do;
           x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                     '('sta') unexpected connstat:' CONN_STAT);
           mtsta.mtact=0;
         end;
         when CONN_STAT='Receiving only' |,
              CONN_STAT='Trying to open' then do;
           x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                     '('sta') unexpected connstat:',
                     CONN_STAT '-> now dead');
           mtsta.mtact=1;
         end;
         otherwise do;
           x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                     '('sta') unknown connstat:',
                     CONN_STAT '-> now dead');
           mtsta.mtact=1; /* now officially declared dead */
         end;
       end/*select*/
     end/*when sta=1090*/;

#switch USE_SRE
#case YES
      /* SRE: S-REP Engine (structured representation engine) **********/
      when sta=2000 then do; /* SRE: listening */
        CONNSTAT(gcn);
        select;
          when CONN_STAT='Listening' then do;
            listencount=listencount+1;
            /* nothing... */
          end;
          when CONN_STAT='Connected' then do;  /* we have contact! */
            /* fork() CMS style */
    x= startserver(mttyp.mtact, mtprt.mtact, mtsel1.mtact, mtsel2.mtact);
            x= logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                 'connected to' foreign_addr':'foreign_port);
            listencount= listencount+1;
            mtsta.mtact= 2010;
            mtttw.mtact= TTW;
            mtfad.mtact= foreign_addr;
```

```
          dosleep=0;
        end;
        when CONN_STAT='Closed' |,
             CONN_STAT='Non-existant' then do;
          x= logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                     '('sta') unexpected connstat:' CONN_STAT);
          mtsta.mtact=0;
        end;
        when CONN_STAT='Trying to open' then do;
          /* nothing */
        end;
        when CONN_STAT='Receiving only' |,
             CONN_STAT='Sending only' then do;
          x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                    '('sta') unexpected connstat:',
                    CONN_STAT '-> now dead');
          mtsta.mtact=1;
        end;
        otherwise do;
          x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                    '('sta') unknown connstat:',
                    CONN_STAT '-> now dead');
          mtsta.mtact=1; /* now officially declared dead */
        end;
      end/*select*/
    end/*when sta=2000*/;

    when sta=2010 then do; /* SRE: connected */
      CONNSTAT(gcn);
      select;
        when CONN_STAT='Listening' then do;
          x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                    '('sta') unexpectedly listening!');
          mtsta.mtact=2000;
        end;
        when CONN_STAT='Connected' then do;
          mtttw.mtact=mtttw.mtact-1;
          yy= 0;
          if bytes_to_read>0 then do;
            mtttw.mtact=mtttw.mtact+TTW;
            tcpreceive(gcn);
tl0=TCPLINE.0;
say 'gcn='gcn'   TCPRC='TCPRC'   TCPLINE.0='TCPLINE.0   'tl0='tl0,
    'bytes_to_read='bytes_to_read
            if TCPRC      = 0           &,
               tl0       /= 'TCPLINE.0' &,
               tcpline.0 >= 1           then do;
              do i=1 to tcpline.0;
                say 'tcpline.'i'='tcpline.i
              end;
              dosleep=0;
            end;
          end;
        end;
        when CONN_STAT='Closed' |,
             CONN_STAT='Non-existant' then do;
          x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                    '('sta') unexpected connstat:' CONN_STAT);
          mtsta.mtact=0;
```

```
        end;
        when CONN_STAT='Receiving only' |,
             CONN_STAT='Sending only' then do;
          x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                    '('sta') unexpected connstat:',
                    CONN_STAT '-> now dead');
          mtsta.mtact=1;
        end;
        when CONN_STAT='Trying to open' then do; /* nothing */ end;
        otherwise do;
          x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                    '('sta') unknown connstat:',
                    CONN_STAT '-> now dead');
          mtsta.mtact=1; /* now officially declared dead */
        end;
      end/*select*/
    end/*when sta=2010*/;
#endswitch

    otherwise do; /* unknown slot status */
      x=logtext('['mtact']' mttyp.mtact'('mtprt.mtact')',
                '('sta') unknown status -> now dead');
      mtsta.mtact=1;
    end;
    end/*select*/

  end/*do mtact=1 to mtmax*/;

  /* now check if we should commit suicide: */
  if (deadcount>100) then do;       /* so many dead slots? */
    x=logtext('***** too many dead slots! deadcount=',
              deadcount 'leaving *****');
    leave;
  end;
  if (listencount<=0 & emptycount+deadcount>=mtmax) then do;
    x=logtext('***** no-one is listening! listencount=',
              listencount 'leaving *****');
    leave;
  end;
  if mtmax>1 & mtsta.mtmax=0 then do;
    mtmax=mtmax-1;
    say 'mtmax is now down to' mtmax
  end;

#switch SRV_WAKEUP_SYSTEM
#case YES
----> this mode is invalid, select either YWAKEUP, WAKEUP or NO
#case YWAKEUP
  if dosleep then do;
     'YWAKEUP (*MSG SMSG NOWAIT'
     select;
       when ywakeup.1='NOWAIT' then do; /* nothing */ end;
       when ywakeup.1='*MSG' then do;
#        do i=0 to ywakeup.0;
#          say 'ywakeup.'i'='ywakeup.i
#        end;
         if ywakeup.5='SMSG' & ywakeup.4='SYS-REACCESS' then do;
           x= reaccess();
           'MSG' ywakeup.3 'I''ve reaccessed my mini disks';
```

```
        end;
      end;
      otherwise do; /* nothing */ end;
    end;
  end;
#case WAKEUP
  if dosleep then do;
    q=queued();
    'WAKEUP +0 (IUCVMSG'
    ret=rc;
    do queued()-q;
      pull text
      select;
        when ret=5 then do;
          parse var text '*' wakeup1 wakeup3 wakeup4
          select;
            when wakeup1='NOWAIT' then nop;
            when wakeup1='MSG' then nop;
            when wakeup1='SMSG' & wakeup4='SYS-REACCESS' then do;
              x= reaccess();
              'MSG' wakeup3 'I''ve reaccessed my mini disks';
            end;
            otherwise do; /* nothing */ end;
          end/*select*/;
        end;
        otherwise do; /* nothing */ end;
      end/*select*/;
    end/* do queued()-q */;
  end;
#endswitch

end/*do forever*/;
x=logtext('***** stopping *****')

'NUCXDROP REXTCPIP';
exit(0);
```

3.1.3 startserver

```
/* ------------------------------------------------------------------- */
/* start a protocol handler of a particular type on a given port       */
/* RETURNS: 0 .. no server started, 1 .. server started                */
startserver: parse arg type, port, sel1, sel2
say 'startserver type='type 'port='port 'sel1='sel1 'sel2='sel2

if port <= 0 then return 0;
xsta= 1;

select;
  when type='GOPHER' | type='FINGER' | type='COOKIE' then do;
    xsta= 1000;
  end;
  when type='SRE' then do;
    xsta= 2000;
  end;
  otherwise return 0;
end;

do i=1 to mtmax;
```

```
  if mtsta.i=0 then leave;
end;
if i=mtmax & mtsta.i/=0 then i= i+1;
if i>mtmax then do;     /* create a new slot */
  mtmax= i;
end;

mtgcn.i= LISTEN(port,0);
mttyp.i= type;
mtprt.i= port;
mtsel1.i= sel1;
mtsel2.i= sel2;
x= logtext('['i']' type'('port') started; TCPRC='TCPRC);
if TCPRC=0 then mtsta.i= xsta; else mtsta.i= 1;
return 1;
```

3.1.4 gopherselector

```
/* -------------------------------------------------------------------- */
gopherselector: parse arg nr full_selector
x= logtext('['nr']' mttyp.nr'('mtprt.nr') sel='full_selector);

if mttyp.nr='EXTRA' then do
  x= mtsel1.nr;
  if full_selector/='' && x/='' then x= x||TAB;
  full_selector= x||full_selector||mtsel2.nr;
end;

parse var full_selector selector':'lfrom':'lto
if lfrom='' then lfrom= 0;
if lto  ='' then lto  = 32700;   /* catch all */

if mttyp.nr='FINGER' then do
  if selector='' then selector='NAMES'
  return gopherspecial(nr 'SPEC-FINGER' selector)
end;

/* empty selector means, return root index */
if selector='' then do
  return gophersend(nr lfrom lto rootindex)
end;

/* fast access to files */
if left(selector,5)='FILE-' then do;
  parse var selector fn'.'ft'.'.;
  q= queued();
  address '' 'LISTFILE' fn ft '*' '(LIFO ALLOC NOHEAD'
  q= queued()-q;
  use= 0;
  do i=1 to q;
    parse pull . . xfm . . xsiz .
    xuse= check_disk_group(xfm, mtfad.mtact);
    if xuse>use then do;
      use= xuse;        /* mark the priority of this disk item  */
      fm= xfm;          /* record the file mode for later       */
    end;
  end;
  if use>0 then return gophersend(nr lfrom lto fn ft fm);
end;
```

```
/* embedded applications */
if left(selector,3)='EA-' then do
  return gopher_eai(nr substr(selector,4))
end;

/* system items */
if left(selector,4)='SYS-' then do
  return gophersys(nr selector)
end;

/* special items */
if left(selector,5)='SPEC-' then do
  return gopherspecial(nr selector':'lfrom':'lto);
end;

/* according to gopher specs: optionally, a date specification can     */
/* follow the selector string, seperated by a tab character.           */
/* ... trash this thing, if present.                                   */
parse var selector selector '05'x .
selector= strip(selector);

/* go down the list of known selectors and see if we care ... */
do i=1 to selcnt;
  parse upper var selfil.i fn'.'ft'.'fm
  parse upper var selector sfn'.'sft'.'sfm

  use= check_group(selgrp.i, mtfad.mtact);

  /* check for wildcards in the filename */
  if (use & fn='*' & ft=sft) then do;
    say 'wildcardfile: selector='selector 'fnm='fn sft fm
    return gophersend(nr lfrom lto sfn ft fm);
  end;

  /* check for wildcards in the filetype */
  if (use & ft='*' & fn=sfn) then do;
    say 'wildcardfile: selector='selector 'fnm='fn sft fm
    return gophersend(nr lfrom lto fn sft fm);
  end;

  if (use & selid.i=selector) then do;
    parse upper var selfil.i fn'.'ft'.'fm
    return gophersend(nr lfrom lto fn ft fm);
  end;
end;

mtmsg.nr= '3 sorry... this item is unknown here';
return 1;
```

3.1.5 gophersend

```
/* ------------------------------------------------------------------ */
gophersend: parse arg nr lfrom lto fn ft fm

/* check out irregularities */
if lfrom<0     then lfrom=0;
if   lto>32500 then lto=32500;
if lfrom>lto   then do; lfrom=0; lto=32500; end;
```

```
upper fn ft fm;
'LISTFILE' fn ft fm '(ALLOC LIFO NOHEAD'
if rc/=0 then do;
  mtmsg.nr= '3 sorry... this item is currently unavailable';
  return 1;
end;
pull . . . . . filesize .
/* say 'send:' fn ft fm 'size='filesize */
if filesize<=0 then do;
  mtmsg.nr= '3 sorry... this item is no ok';
  return 1;
end;

mtfil.nr= fn ft fm;
if lto  <filesize then mtsiz.nr= lto;   else mtsiz.nr= filesize;
if lfrom>0        then mtrec.nr= lfrom; else mtrec.nr= 0;
return 2;
```

3.1.6 gopherspecial

```
/* ------------------------------------------------------------------ */
gopherspecial: parse arg nr sel':'lfrom':'lto
xfnm= 'GOPHERDD TMP'nr rwdisk
address command 'STATEW' xfnm
if rc=0 then 'ERASE' xfnm;
qtxt= 'QUERY LOGMSG'
select;
  when left(sel,11)='SPEC-COOKIE' then do;
    'COOKIE' xfnm
    return gophersend(nr 0 32700 xfnm);
  end;
  when word(sel,1)='SPEC-FINGER' then do;
    parse var sel . sel
    'FINGERXX' xfnm '(0' sel
    return gophersend(nr 0 32700 xfnm);
  end;
  when substr(sel,1,11)='SPEC-FINGER' /* &,
       substr(sel,12,1)='05'x */ then do;
    'FINGERXX' xfnm '(0' substr(sel,13);
    return gophersend(nr 0 32700 xfnm);
  end;
  otherwise do;
    parse var sel s1||'05'x||s2
    if s1='SPEC-THUMB' & s2/='' then do;
      'FINGERXX' xfnm '(1' s2
      return gophersend(nr 0 32700 xfnm);
    end;
    mtmsg.nr= '3 sorry... this SPEC-item is unknown here';
    return 1;
  end;
end;

return 0;
```

3.1.7 gophersys

```
/* ------------------------------------------------------------------ */
gophersys: parse arg nr sel
```

```
xfnm= 'GOPHERDD TMP'nr rwdisk
address command 'STATEW' xfnm
if rc=0 then 'ERASE' xfnm;
qtxt= 'QUERY LOGMSG'
select;
  when left(sel,10)='SYS-QUERY ' then qtxt='QUERY' substr(sel,10);
  when left(sel,8)='SYS-STAT' then do;
    push 'data-to=Gopher@awiwuw11.wu-wien.ac.at'
    push 'links-to=Gopher@awiwuw11.wu-wien.ac.at'
    push 'bugs-to=Gerhard.Gonter@wu-wien.ac.at'
    push 'contact=#<CL_CONTACT>#'
    'IDENTIFY (LIFO';
    parse pull z
    push 'server='z
    push 'revision='REVISION
    push 'note=Implemented using #<SRV_TCP># by',
               '<Gerhard.Gonter@wu-wien.ac.at>'
    push 'note=#<SRV_ORGANIZATION>#'
    push 'note=This is a VM/CMS GOPHER Server of the',
    'EXECIO 9 DISKW' xfnm '(FINIS'
    return gophersend(nr 0 32700 xfnm);
  end;
  when selector='SYS-REACCESS' then do;
    x= reaccess();
    mtmsg.nr= '3 I''ve reaccessed my mini disks';
    return 1;
  end;
  when sel='SYS-STOP' then do;
    say 'stopping...'
    /* stopflag=1; */
    mtmsg.nr= '3 I''m *not* stopping now, honey...';
    return 1;
  end;
  otherwise do;
    mtmsg.nr= '3 sorry... this SYS-item is unknown here';
    return 1;
  end;
end;

'MAKEBUF'
upper qtxt
'EXECIO * CP (FIFO STR' qtxt
'EXECIO' queued() 'DISKW' xfnm '(FINIS'
'DROPBUF'
return gophersend(nr 0 32700 xfnm);
```

3.1.8 gopher_eai

```
/* ------------------------------------------------------------------- */
gopher_eai: parse arg nr selector '05'x keyword

xfnm= 'GOPHERDD TMP'nr rwdisk
address command 'STATEW' xfnm
if rc=0 then 'ERASE' xfnm;

do i=1 to eaicnt;
  use= check_group(eaigrp.i, mtfad.mtact);
  if (use & eaiid.i=selector) then do;
    cmdline= prepare_command(eaipgm.i, xfnm, keyword);
```

```
say 'cmd:' cmdline
    interpret cmdline;
    return gophersend(nr 0 32700 xfnm);
  end;
end;
mtmsg.nr= '3 sorry... this item is unknown here';
return 1;
```

3.1.9 reaccess

```
/* ------------------------------------------------------------------- */
reaccess: parse arg .
x=logtext('*** reaccessing disks');
'REACCESS'
x= readselectors(selectorfile);  /* initialize selector list          */
x= readgroups(groupfile);        /* initialize group list             */
x= read_eai(eaifile);            /* initialize EAI handler list       */
x= read_disk_groups(dskgrpfile); /* initialize list of disk groups    */

address '' 'STATE GSH_REAC EXEC'
if rc=0 then do;
  say 'executing gopher server reaccess hook code'
  'GSH_REAC'
end;

address '' 'STATE GSH_WAIT EXEC'
if rc=0 then gsh_wait_exists= 1;
else gsh_wait_exists= 0;

return 0;
```

3.1.10 readselectors

```
/* ------------------------------------------------------------------- */
readselectors: parse arg fn ft fm
upper fn ft fm;
'LISTFILE' fn ft fm '(ALLOC LIFO NOHEAD'
if (rc/=0) then return -1;
pull . . . . . filesize .
say 'selector file has' filesize 'entries'
selcnt=0;
selid.=''
selfil.='';
selgrp.='';
do i=1 to filesize;
  'EXECIO 1 DISKR' fn ft fm '(LIFO'
  parse pull fil . grp sel
/*say '['i'] ' fil sel  */
  if (fil/='*' & fil/='#') then do;
    selcnt=selcnt+1;
    selid.selcnt= strip(sel);
    selfil.selcnt= fil;
    selgrp.selcnt= grp;
  end;
end/*do*/;
x=logtext('***' selcnt 'selectors found');
'FINIS' fn ft fm;
return 0;
```

3.1.11 readgroups

```
/* ------------------------------------------------------------------- */
readgroups: parse arg fn ft fm
upper fn ft fm;
address '' 'LISTFILE' fn ft fm '(ALLOC LIFO NOHEAD'
if (rc/=0) then return -1;
pull . . . . . filesize .
say 'group file has' filesize 'entries'
grpcnt= 0;
grpid.= ''
grpaddr.= '';
do i=1 to filesize;
  'EXECIO 1 DISKR' fn ft fm '(LIFO'
  parse pull grp addrs
/*say '['i'] ' fil sel  */
  if (grp/='*' & grp/='#') then do;
    do while addrs/='';
      parse var addrs addr addrs
      grpcnt= grpcnt+1;
      grpid.grpcnt= grp;
      grpaddr.grpcnt= addr;
    end/*while*/
  end;
end/*do*/;
x= logtext('***' grpcnt 'groups found');
'FINIS' fn ft fm;
return 0;
```

3.1.12 read_eai

```
/* ------------------------------------------------------------------- */
read_eai: parse arg fnm
upper fnm;
'LISTFILE' fnm '(ALLOC LIFO NOHEAD'
if (rc/=0) then return -1;
pull . . . . . filesize .
say 'EAI file has' filesize 'entries'
eaicnt= 0;
eaiid.= ''
eaipgm.= '';
eaigrp.= '';
do i=1 to filesize;
  'EXECIO 1 DISKR' fnm '(LIFO'
  parse pull hdl . grp pgm
  say '['i'] ' hdl grp pgm
  if (hdl/='*' & hdl/='#') then do;
    eaicnt= eaicnt+1;
    eaiid.eaicnt= strip(hdl);
    eaipgm.eaicnt= pgm;
    eaigrp.eaicnt= grp;
  end;
end/*do*/;
x= logtext('***' eaicnt 'EAI handlers found');
'FINIS' fnm;
return 0;
```

3.1.13 read_disk_groups

```
/* ------------------------------------------------------------------- */
read_disk_groups: parse arg fnm
upper fnm;
address '' 'LISTFILE' fnm '(ALLOC LIFO NOHEAD'
if (rc/=0) then return -1;
pull . . . . . filesize .
say 'disk group file has' filesize 'entries'
dskgrpcnt= 0;
dskgrpnam.= ''
dskgrpdsk.= '';
do i=1 to filesize;
  'EXECIO 1 DISKR' fnm '(LIFO'
  parse pull dsk pri grps
/*say '['i'] ' fil sel  */
  if (dsk/='*' & dsk/='#') then do;
    do while grps/='';
      parse var grps grp grps
      dskgrpcnt= dskgrpcnt+1;
      dskgrpdsk.dskgrpcnt= dsk;
      dskgrppri.dskgrpcnt= pri;
      dskgrpnam.dskgrpcnt= grp;
    end/*while*/
  end;
end/*do*/;
x= logtext('***' dskgrpcnt 'disk groups found');
'FINIS' fnm;
return 0;
```

3.1.14 xconnstat

```
#switch SRV_TCP
#case REXTCPIP
/* ------------------------------------------------------------------- */
xconnstat: parse arg gcn msg
say msg
connstat(gcn);
say 'conn_stat='conn_stat'   bytes_to_read='bytes_to_read'  ',
    'unacked_bytes='unacked_bytes
say 'connection:' local_addr':'local_port '  -> ',
    foreign_addr':'foreign_port
return 0;
#endswitch
```

3.1.15 logtext

```
/* ------------------------------------------------------------------- */
logtext: parse arg txt
'EXECIO 1 DISKW' logfile '(FINIS STRING' date(sorted) time() txt
say date(sorted) time() txt
return rc;
```

3.1.16 prepare_command

```
/* ------------------------------------------------------------------- */
prepare_command: parse arg template, tmpfn tmpft tmpfm, keyw
outline= '';
ll= length(template);
```

```
j= 1;
do while j<=ll;
  select;
    when substr(template, j, 3)='@fn' then do;
      outline= outline||tmpfn;
      j= j+3;
    end;
    when substr(template, j, 3)='@ft' then do;
      outline= outline||tmpft;
      j= j+3;
    end;
    when substr(template, j, 3)='@fm' then do;
      outline= outline||tmpfm;
      j= j+3;
    end;
    when substr(template, j, 3)='@kw' then do;
      outline= outline||keyw;
      j= j+3;
    end;
    otherwise do;
      outline= outline||substr(template, j, 1);
      j= j+1;
    end;
  end/*select*/;
end/*do*/;
return outline;
```

3.1.17 check_group

```
/* ------------------------------------------------------------------- */
check_group: parse arg xxgrp, fad
if (xxgrp='*') then return 1;

do j=1 to grpcnt;
  if (grpid.j=xxgrp &,
      grpaddr.j=substr(fad,1,length(grpaddr.j)) )
    then return 1;
end;/*do*/

return 0;
```

3.1.18 check_disk_group

```
/* ------------------------------------------------------------------- */
check_disk_group: parse arg xxfm, fad
do j=1 to dskgrpcnt;
  if dskgrpdsk.j=xxfm then do;
    u= check_group(dskgrpnam, mtfad.mtact);
    if u>0 then return dskgrppri.j;
  end;
end;
return 0;
```

3.1.19 EBCDIC to ASCII conversion

```
/* E2A EXEC simple ebcdic to ascii translate by JCA ------------------ */
E2A:
   return translate(arg(1), ,
```

```
    '000102030009007F0000000B0C0D0E0F'x||,
    '101112130000080018190001C1D1E1F'x||,
    '00000000000A171B0000000000050607'x||,
    '0000160000000004000000001415001A'x||,
    '200000000000000000000002E3C282B7C'x||,
    '2600000000000000000021242A293B7E'x||,
    '2D2F000000000000000002C255F3E3F'x||,
    '005E000000000000000603A2340273D22'x||,
    '006162636465666768690007B00000000'x||,
    '006A6B6C6D6E6F707172007D00000000'x||,
    '007E737475767778797A0000005B0000'x||,
    '0000000000000000000000000005D0000'x||,
    '7B414243444546474849000000000000'x||,
    '7D4A4B4C4D4E4F505152000000000000'x||,
    '5C005354555657585950A000000000000'x||,
    '303132333435363738390000000000000'x);
```

3.1.20 ASCII to EBCDIC conversion

```
/* A2E EXEC simple ascii to ebcdic translate by JCA ------------------ */
A2E:
   return translate(arg(1), ,
     '00010203372D2E2F1605250B0C0D0E0F'x||,
     '101112133C3D322618193F271C1D1E1F'x||,
     '405A7F7B5B6C507D4D5D5C4E6B604B61'x||,
     'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F'x||,
     '7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6'x||,
     'D7D8D9E2E3E4E5E6E7E8E9ADE0BD716D'x||,
     '79818283848586878889919293949596'x||,
     '979899A2A3A4A5A6A7A8A9C04FD05F07'x||,
     '00010203372D2E2F1605250B0C0D0E0F'x||,
     '101112133C3D322618193F271C1D1E1F'x||,
     '405A7F7B5B6C507D4D5D5C4E6B604B61'x||,
     'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F'x||,
     '7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6'x||,
     'D7D8D9E2E3E4E5E6E7E8E9ADE0BD716D'x||,
     '79818283848586878889919293949596'x||,
     '979899A2A3A4A5A6A7A8A98B4F9B5F07'x);



# endswitch
# this ends the block of fixes for the server modules


# This is a block of definitions for the client code.
# switch MAKE_CLIENT
# case YES
```

4 Client

4.1 Module GOPHERC EXEC

```
#erase GOPHERC EXEC
#module GOPHERC EXEC
/**********************************************************************/
/* File GOPHERC EXEC #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME># */
/*                                                                    */
```

```
/* The main module of the GOPHER Client for VM/CMS                  */
/*                                                                  */
/* written:      1992 02 26: <Gerhard.Gonter@wu-wien.ac.at>         */
/* latest update: 1993-05-26                                        */
/********************************************************************/
parse arg host port ity sel

/* address command ... not yet ... */
/* @@@ 'GLOBALVV SELECT GOPHER STACK HOST' @@@ */
address '' 'GLOBALV SELECT GOPHER STACK HOST'
parse pull dhost .
/* @@@ 'GLOBALVV SELECT GOPHER STACK SEL' @@@ */
address '' 'GLOBALV SELECT GOPHER STACK SEL'
parse pull dsel .
'GLOBALV SELECT GOPHER STACK ITYPE PORT DNAME'
pull dity .
pull dport .
parse pull dname

/* use default values from lasting globalv */
if host='' | host='=' then host= dhost
if port='' | port='=' then port= dport
if ity='' then ity= dity
if sel='' then sel= dsel

/* if nothing is specified use preset values: */

/* to be configured at installation time by ROSE:  - - - - - - - - - - */
if host='' | host='=' then host= #<CL_DEFAULT_HOST>#;
if port='' | port='=' then port= #<CL_DEFAULT_PORT>#;
if ity='' then ity=#<CL_DEFAULT_TYPE>#;  /* default: directory */
/* - - - - - - - - - - - - - - - - - - -  end of configuration section */

select;
  when ity='0' then xity='<F>';
  when ity='1' then xity='<D>';
  when ity='7' then xity='<S>';
  when ity='B' then xity='<B>'; /* BSS encoded binary data */
  otherwise xity='>'ity'<';
end;

if dname='' then dname= xity substr('#<STARTUP_DIR>#',1,76);

/* @@@ 'GLOBALXV SELECT GOPHER SETLP TMPCNT 1'                    @@@ */
/* @@@ 'GLOBALXV SELECT GOPHER SETLP GHOST' host':'port '<'ity'>' sel  */
/* @@@ 'GLOBALXV SELECT GOPHER SETLP GI1:TMP0' dname;             @@@ */
/* @@@ 'GLOBALXV SELECT GOPHER SETLP GI2:TMP0' ity host port sel   @@@ */
address '' 'GLOBALV SELECT GOPHER SETL TMPCNT 1'
address '' 'GLOBALV SELECT GOPHER SETL GHOST' host':'port '<'ity'>' sel
address '' 'GLOBALV SELECT GOPHER SETL GI1:TMP0' dname;
address '' 'GLOBALV SELECT GOPHER SETL GI2:TMP0' ity host port sel

'SET EMSG OFF'

tmpfile='GOPHTMP TMP0 A';

if 'THIS'/='JUNK' then do;
  'GOPHFTCH' tmpfile ity host port sel
  'XEDIT' tmpfile '(PROFILE GOPHER'
```

```
end; else do;
  if ity='1' then linewidth=250;
  else linewidth=80;
  'XEDIT' tmpfile '(WIDTH' linewidth 'PROFILE GOPHER'
  'RECFM V'
  'LRECL' linewidth
  'TRUNC' linewidth
  'SET LINEND OFF'
  ':1'
  'GOPHFTCH' tmpfile'(XEDIT' ity host port sel
  ':1'
  'SET ALT 0 0'
end;

say 'The Internet GOPHER service wishes you a nice and pleasant day.'
say 'We hope we can welcome you back on board of the VieGOPHER system.'
say 'For suggestions, complaints or questions contact'
say '#<CL_CONTACT>#.'
/* @@@ 'ERASE LASTING GLOBALXV' @@@ */
'ERASE GOPHTMP TMP0'
'SET EMSG ON'

exit(0);
```

4.2 Module GOPHER XEDIT

```
#erase GOPHER XEDIT
#module GOPHER XEDIT
/*********************************************************************/
/* File GOPHER XEDIT #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME># */
/*                                                                   */
/* Setup for the GOPHER Client                                       */
/*                                                                   */
/* written:       1992-02-26: <Gerhard.Gonter@wu-wien.ac.at>         */
/* latest update: 1993-05-31                                         */
/*********************************************************************/

parse upper arg modus
/* modus defines a 'prefix' code for alternative keyboard setups     */
/* The idea is from Mary Posey, I herewith steal it :)               */
/* Here are the details:                                             */
/* STD ... (or nothing) means normal operation mode                  */
/* 12K ... normal 12 key setup                                       */
/* RNG ... setup for ring display                                    */
/* 12R ... setup for ring display with only 12 keys                  */
if modus=''| words(modus)>1 then modus='STD';

/* set a few parameters ... */
'SET NUMBER ON'
'SET NULLS ON'
'SET CURLINE ON 3'
'SET SCALE OFF'
'SET TOFEOF ON'
'SET CMDLINE TOP'
'SET PREFIX OFF'
'SET HEX ON'
'SET CASE MIXED IGNORE'
/* ':1' ... this should only be done when current line is 0 */
```

```
/* install the gopher specific keys */
'SET PF7  command backward'
'SET PF8  command forward'
'SET PF10 command msg undefined PF key!'

select;
  when modus='12K' | modus='12R' then do;
    'SET PF1  command xedit GOPHTMP TMP0 (PROFILE GOPHER'
    'SET PF3  cancel'
    'SET PF4  command msg undefined PF key!'
    'SET PF5  command macro gophscrn split'
    'SET PF6  command macro gophscrn join'
    'SET PF9  command macro gophring'
    'SET PF11 command exec gophstp'
    'SET PF12 ignore command cursor cmdline 1 priority 30'
  end;
  when modus='STD' | modus='RNG' then do;
    'SET PF1  command help gopher'
    'SET PF3  command macro gophxqit'
    'SET PF4  command macro gophxbmi'
    'SET PF5  command macro gophxbmc'
    'SET PF6  command macro gophxbms'
    'SET PF9  command macro gophxsvi'
    'SET PF12 command xedit'
  end;
  otherwise nop;
end/*select*/;

'SET PF13 command xedit GOPHTMP TMP0 (PROFILE GOPHER'
'SET PF14 command macro gopher STD'
'SET PF15 cancel'
'SET PF16 command msg undefined PF key!'
'SET PF17 command macro gophscrn split'
'SET PF18 command macro gophscrn join'
'SET PF19 command backward'
'SET PF20 command forward'
'SET PF21 command macro gophring'
'SET PF22 command msg undefined PF key!'
'SET PF23 command exec gophstp'
'SET PF24 ignore command cursor cmdline 1 priority 30'

select;
  when modus='STD' then do;
    'SET PF2  command macro gopher 12K'
    'SET PF11 command macro gophxftc';
    'SET ENTER command macro gophxftc'
  end;
  when modus='12K' then do;
    'SET PF2  command macro gopher STD'
  end;
  when modus='RNG' then do;
    'SET PF2  command macro gopher 12R'
    'SET PF11 command macro gophrng3';
    'SET ENTER command macro gophrng3';
  end;
  when modus='12R' then do;
    'SET PF2  command macro gopher 12R'
  end;
```

```
  otherwise nop;
end/*select*/

/* define possible function key display strings */
if modus='12K'|modus='12R' then do;
T5L1=' 1= First     2=(basepfs) 3= Exit      4=         ',
       ' 5= Split       6= Join ';
T5L0=' 7= PageUP    8= PageDN   9= ShowRing 10=         ',
     '11= Setup      12= Execute';
end; else do;
T5L1=' 1= Help      2=(altpfs)  3= GoBack  4= Disp->BMK ',
       ' 5= Curs->BMK   6= BMK  ';
T5L0=' 7= PageUP    8= PageDN   9= Save   10=           ',
     '11= Display    12= Circle';
end;

/* detailed function key descriptions */
/***************
T9L1=' 1= Help      2=           3= GoBack  4= Disp->BMK ',
     ' 5= Curs->BMK   6= BMK  ';
T9L2=' 7= PageUP    8= PageDN    9= Save   10=           ',
     '11= Display    12= Circle';
*****************/
T9L3='13= First    14=          15= Exit     16=         ',
     '17= Split      18= Join ';
T9L4='19= PageUP   20= PageDN   21= ShowRing 22=         ',
     '23= Setup      24= Execute';
T9X1='<F> are files, <D> are menus';
T9X2='place the cursor in the line of an interesting item and press ENTER';

/* display extra information on the screen */
'EXTRACT /LSCREEN/SCREEN/FTYPE/';
pl0=LSCREEN.1;
/* @@@ 'GLOBALXV SELECT GOPHER STACK GHOST' @@@ */
address '' 'GLOBALV SELECT GOPHER STACK GHOST'
parse pull ghost
/* @@@ 'GLOBALXV SELECT GOPHER STACK GI1:'FTYPE.1 @@@ */
address '' 'GLOBALV SELECT GOPHER STACK GI1:'FTYPE.1
parse pull dirname

/* default info level */
'GLOBALV SELECT GOPHER STACK INFOLEVEL'
parse pull infolevel
if infolevel='' then infolevel=#<CL_INFO_LEVEL>#;
if words(screen.1)>3 then infolevel=0;

select;
  when infolevel>=1 & infolevel<=3 then do;
  end;
  when infolevel=4 | infolevel=5 then do;
    'SET RESERVED' PL0-1 'BLUE NONE NOHIGH' T5L1
    'SET RESERVED' PL0   'BLUE NONE NOHIGH' T5L0
  end;
  when infolevel=6 then do;
  end;
  when infolevel=7 | infolevel=8 then do;
  end;
  when infolevel=9 then do;
    'SET RESERVED 5 RED NONE HIGH' T9X1
```

```
    'SET RESERVED 6 RED NONE HIGH' T9X2
    'SET RESERVED' PL0-3 'BLUE NONE NOHIGH' T5L1
    'SET RESERVED' PL0-2 'BLUE NONE NOHIGH' T5L0
    'SET RESERVED' PL0-1 'BLUE NONE NOHIGH' T9L3
    'SET RESERVED' PL0   'BLUE NONE NOHIGH' T9L4
  end;
  otherwise nop; /* especially when infolevel=0 then */
end/*select*/;

select;
  when modus='RNG' | modus='12R' then do;
    'SET RESERVED 3 BLUE NONE NOHIGH list of currently active items; '||,
       'select one ...'
  end;
  otherwise do;
    'SET RESERVED 3 BLUE NONE NOHIGH' substr(dirname,1,79);
    'SET RESERVED 4 BLUE NONE NOHIGH' substr(ghost,1,79);
  end;
end/*select*/
```

4.3 Module GOPHER HELPCMS

```
#erase GOPHER HELPCMS
#module GOPHER HELPCMS
How to Use the VM Gopher at San Jose State
------------------------------------------------------------------------

***  Please note: this file was written by John Sroka, San Jose State   ***
***  University, and reflects the local installation there, most parts  ***
***  of this text, however, can be applied for other installations. -GG ***

The VM gopher client/server at San Jose State University allows you
to browse information files on the campus, or at any of the (175 and
counting) other gopher sites around the world.

Gopher was developed at the University of Minnesota.  The CMS version
was written by Gerhard Gonter at the University of Economics at Vienna,
Austria (Gerhard.Gonter@wu-wien.ac.at, +43/1/31336/4111).

Please direct questions or comments about the San Jose State VM gopher
to John Sroka at Information Systems & Computing (sroka@sjsuvm1.sjsu.edu)
phone (408) 924-2332.


*** The Display ***

A typical gopher display simply contains a list of information files,
or directories which list more information files.  The first character
in the descriptor tells you what the item is:

<F>  an information file
<D>  a directory of more information files
<S>  a text database where searching can be done
# *** SJSU ***
# <E>  a exec call (runs an external program)
<?>  an item currently unavailable to VM gopher
***  a line of information just on the screen
```

```
*** The PF Keys ***

You use PF keys to perform most of the gopher functions:

PF1  calls this help file (main display only)
PF3  'quits' to the previous screen, if at main display will exit gopher
PF4  adds current display to your bookmark file (see below)
PF5  adds item cursor is positioned on to your bookmark file (see below)
PF6  retrieves your cumulative bookmark item list (from PF4 and PF5)
PF7  page up ('back') in the current file
PF8  page down ('forward') in the current file
PF9  appends current display or item list to GOPHER SAVEFILE A
# *** SJSU ***
# PF10 spools the displayed item to your virtual printer
PF11 displays item at current cursor position
PF12 'jumps' to next available gopher screen, unless at main menu

*** Using VM Gopher ***

# *** SJSU ***
# VM gopher may be initialized by either typing INFO or VMGOPHER at any
# CMS command line.
After invoking the VieGOPHER client, you will be
presented with the 'Main Menu' from which
you may select (with the cursor control and PF11 keys) a file or direct-
ory to display.  Bear in mind that some of the items must be 'gophered'
from halfway around the world, so allow a few seconds after pressing
the PF11 key for this process to happen.   Occasionally a server may
be unavailable, in which case no information items will be displayed.
Simply press PF3 to drop to the previous screen and try again later.

As you get 'deeper' into the gopher files, you may want to refer back
to a previous screen, just press the PF12 (Next) key to toggle through
the previous screens.  Use the PF8 and PF7 keys to browse up and down
the files, if you see an item you want to keep for later processing,
pressing PF9 will append the current item onto file GOPHER SAVEFILE
on your A disk.
# *** SJSU ***
# The Print key (PF10) will route the displayed item
# to your virtual printer.  See the CIC Bulletin Board item within
# Computing and Network Services for information on setting up your
# virtual printer.

*** About Bookmarks ***

This function allows you to build a customized screen of information
files culled from your browsings through gopher.  This is very handy if
you find an information source you want to refer to often.  To add
an item to your customized list, simply press PF4 to add the currently
displayed file to the list, or PF5 to add the item at the current
cursor position.  In the latter case, the cursor must be positioned
on a valid file or directory descriptor.  To retrieve your customized
list, just press the PF6 (Bookmark) key.

Note there are no keys to remove saved bookmarked items.  Simply exit
GOPHER and manually edit (XEDIT GOPHER BOOKMARK A) out the unwanted
entries.

*** Adding Information to Gopher ***
```

```
We encourage departments who wish to publish information on the gopher
to contact us at the e-mail address above.   If you've a Macintosh
or Unix box which is attached to the Internet, you may wish to build
your own server, or the information may be added to the VM gopher.

Your comments and questions are also welcomed.
```

4.4 Module GOPHFTCH EXEC

```
#erase GOPHFTCH EXEC
#module GOPHFTCH EXEC
```

4.4.1 main

```
/**********************************************************************/
/* File GOPHFTCH EXEC  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME># */
/*                                                                    */
/* Fetch an item from a GOPHER server                                 */
/*                                                                    */
/* Uses: REXTCPIP MODULE or RXSOCKET MODULE                           */
/*                                                                    */
/* written:       1992-02-28: <Gerhard.Gonter@wu-wien.ac.at>          */
/*                1992-07-14: implemented type=w for search           */
/*                1993-03-21: redesign/parameter exchance via STACK   */
/*                1993-04-21: support for RXSOCKET V2                 */
/* latest update: 1993-05-26                                         */
/**********************************************************************/
parse arg fn ft fm p1 p2

if fn='' then exit(28);

/* get the GOPHER coordinates for item to fetch */
select;
  when p1='PULL' then do; /* gopher co-ordinates on stack */
    rsel='';
    do forever;
      parse pull aname'='avalue
      upper aname;
      if aname='END' then leave;
      select;
        when aname='TYPE' then ty=avalue;
        when aname='HOST' then rsrv=avalue;
        when aname='PORT' then rprt=avalue;
        when aname='PATH' then rsel=rsel||avalue;
        when aname='HTAP' then rsel=avalue||rsel;
        otherwise nop; /* do nothing; unknown attribute */
      end/*select*/
    end;/* do forever */
    XL.0=4;
    XL.4=rsel;
  end;
  when p1='FILE' then do; /* gopher coordinates are in a file */
    'EXECIO 4 DISKR' p2 '(FINIS STEM XL.'
    if XL.0 < 4 then return -1;
    ty=   XL.1;
    rsrv= XL.2;
    rprt= XL.3;
    rsel= XL.4;
```

```
  end;
  otherwise do; /* parameters are in the argument list */
    ty=p1;
    parse var p2 rsrv rprt rsel
    XL.0=4;
    XL.4=rsel;
  end;
end/*select*/;

if ty='' then ty=0;

/* to be configured at installation time by ROSE: - - - - - - - - - - - */
if rsrv='' then rsrv=#<CL_DEFAULT_HOST>#;      /* default server       */
if rprt='' then rprt=#<CL_DEFAULT_PORT>#;      /* default servers port */
tout=#<CL_TIME_OUT>#;                   /* timeout for TCPOPEN (sec)   */

#switch CL_USE_LOCALFETCH
#case YES
localhost='#<CL_LOCALHOST>#';  /* IP where client runs */
localgopher='70';              /* Port where Gopher server listens on  */
selectorfile='GOPHERD SELECTOR';
indexfile='GOPHERD INDEX';
gopherdisk='*';
#endswitch
/* - - - - - - - - - - - - - - - - - - -  end of configuration section */

tab='05'x;
crlf='0D'x || '0A'x;
cr='0D'x;
lf='0A'x;

'SUBCOM XEDIT'
if rc=0 then inXedit=1; else inXedit=0;

CSO_n=0;
if ty='9'||ty='I' then is_text=0;
else is_text=1;

#switch CL_USE_FDNS
#case YES
/* IP address resolution using FTP (see VMGOPHER.DOC for details) ******/
if verify(rsrv,'0123456789.')/=0 then do;
  'FDNS' rsrv '(LIFO'
  pull code . ip .
  if (code/='AD') then do;
    exit(0);
  end;
  rsrv=ip
end;
#endswitch

parse var fm fm'('fm_opt
upper fm_opt;
fnm=fn ft fm
rsrv_name= rsrv;        /* remember the name of the server             */
                        /* rsrv is later on the ip address or some     */
                        /* other data structure containing the         */
                        /* address of the server.                      */
```

```
if fm_opt='' then 'ERASE' fnm;

#switch CL_TCP
#case REXTCPIP
'REXTCPIP';
#switch CL_USE_FDNS
#case NO
if verify(rsrv,'0123456789.')/=0 then do;
  ip= GetIPAddr(rsrv);
  if (ip='') then do;
    'NUCXDROP REXTCPIP'
    x= report(fnm, fm_opt, "can't get" rsrv"'s IP address");
    exit(1001);
  end;
  rsrv=ip
end;
#endswitch
#case RXSOCKET
# V1/V2 1993-04-21
parse value Socket('Initialize', 'Gopher') with x x_rest
if x_rest='' then do;
  RXSOCKV= 1;
  if x>0 then x= 0; /* well... this could cause problems... */
end; else RXSOCKV= 2;
if x/=0 then do;
ERR:
  'NUCXDROP RXSOCKET'
# "EXECIO 1 DISKW" fnm "(FINIS STRING can't start TCP using RXSOCKET"
  x= report(fnm, fm_opt, "can't start TCP using RXSOCKET");
  exit(1001);
end;
if RXSOCKV=1 then sock= Socket('Socket', 'AF_INET', 'Sock_Stream');
if RXSOCKV=2 then do;
  parse value Socket('Socket', 'AF_INET', 'Sock_Stream') with rc sock .
  if rc/=0 then sock= -1;
end;
if sock=-1 then signal ERR;

if verify(rsrv,'0123456789.')=0 then do;
  /* call a converter function */
  ip= INet_Addr(rsrv);  /* doesn't work ... */
end; else do;
  if RXSOCKV=1 then ip= Socket('GetHostByName', rsrv);
  if RXSOCKV=2 then do;
    parse value Socket('GetHostByName', rsrv) with rc ip .
    if rc/=0 then ip=-1;
  end;
# if ip=-1 then ip= GetHostByName(rsrv); Huh???? what's this??
  if ip=-1 then do;
    'NUCXDROP RXSOCKET'
    x= report(fnm, fm_opt, "MSG can't resolve hostname" rsrv);
    exit(1001);
  end;
  rsrv= ip;
end;
#endswitch

#switch CL_USE_LOCALFETCH
#case YES
```

```
/* this is here to fix a problem with REXTCPIP, where   */
/* local connections are not handled propperly...       */
if rsrv_name=localhost & rprt=localgopher then do;
  if fm_opt='PUSH' then push('TOP');
  if fm_opt='QUEUE' then queue('TOP');
  x= localfetch(fnm, fm_opt, ty, rsel);
#switch CL_TCP
#case REXTCPIP
  'NUCXDROP REXTCPIP'
#case RXSOCKET
  'NUCXDROP RXSOCKET'
#endswitch
  if fm_opt='PUSH' then push('BOT');
  if fm_opt='QUEUE' then queue('BOT');
  exit(0);
end;
#endswitch

if (inXedit=1) then do;
  address XEDIT 'MSG trying to open' rsrv_name '(timeout is' tout 'seconds)'
  address XEDIT 'REFRESH'
end;

#switch CL_TCP
#case REXTCPIP
gc= TCPOPEN(rsrv,rprt,tout);
if (gc/=0 & TCPRC=0) then do;
/* say 'rsel=['rsel']' */
  TCPSEND(gc,rsel);
  if ty='2' then TCPSEND(gc,'quit');
#case RXSOCKET
if RXSOCKV=1 then
  rc=Socket('Connect', sock, AF_INET || Htons(rprt) || rsrv);
if RXSOCKV=2 then do;
  parse value Socket('Connect', sock, AF_INET rprt rsrv) with rc .
  if rc/=0 then rc= -1;
end;
if rc/=-1 then do;
  if RXSOCKV=1 then do;
    bytes_out= Socket('Write', sock, E2A(rsel)||crlf);
    if ty='2' then
      bytes_out= bytes_out + Socket('Write', sock, E2A('quit')||crlf);
  end;
  if RXSOCKV=2 then do;
    parse value Socket('Write', sock, E2A(rsel)||crlf) with rc bytes_out
    if ty='2' then do;
      parse value Socket('Write', sock, E2A('quit')||crlf) with rc bytes_inc
      bytes_out= bytes_out + bytes_inc
    end;
  end;
  xline='';
#endswitch

  if (inXedit=1) then do;
    address XEDIT 'MSG please hold on while receiving data...'
    address XEDIT 'REFRESH'
  end;
  receivecnt=0;
  xrececnt=1;
```

```
  if fm_opt='PUSH' then push('TOP');
  if fm_opt='QUEUE' then queue('TOP');

  do forever;
#switch CL_TCP
#case REXTCPIP
    connstat(gc);
    if (conn_stat/='Connected') then leave;
    if (is_text) then TCPRECEIVE(gc,'WAITKB');
    else TCPRECEIVE(gc,'WAITKB',bin);
    select;
      when (TCPRC=0) then do;
        do i=1 to TCPLINE.0;
          line= TCPLINE.i
#case RXSOCKET
    if RXSOCKV=1 then do;
      bytes_read= Socket('Read', sock, 'data');
      if bytes_read=-1 then rc=999; else rc=0;
    end;
    if RXSOCKV=2 then do;
      parse value Socket('Read', sock) with rc bytes_read data
    end;
    if rc/=0 then do;
      x= report(fnm, fm_opt, 'error...');
      'NUCXDROP RXSOCKET'
      exit(0);
    end;
    if bytes_read=0 then leave;
    xline= xline||data;
    do forever;
      if (is_text) then do;
        xli= index(xline,lf);
        if xli=0 then leave;
        line= substr(xline,1,xli-1);
        if substr(line,1,1)=cr then line= substr(line,2);
        ll= length(line);
        if ll>0 then if substr(line,ll)=cr then line= substr(line,1,ll-1);
        line= A2E(line);
        xline= substr(xline,xli+1);
      end; else do;
        line= xline; /* no modification */
      end;
#endswitch
        if (is_text) then do;
          if (line='.') then leave;  /* end of transmission */
          if (substr(line,1,1)='.') then line= substr(line,2);
          x= prclin(fnm, fm_opt, ty, line);
        end; else do; /* binary, just write to file, no processing */
          x= emmit(fnm, fnm_opt, line);
        end;
        receivecnt= receivecnt+1;
        if (inXedit=1 &,
            ((receivecnt  =16)|,
             (receivecnt  =32)|,
             (receivecnt  =64)|,
             (receivecnt//128=0))) then do;
          xrececnt= xrececnt*2;
          if fm_opt='XEDIT' then address XEDIT ':1'
```

```
          address XEDIT 'MSG please hold on; receiving more data:',
                         receivecnt 'lines sofar...';
          address XEDIT 'REFRESH';
          if fm_opt='XEDIT' then address XEDIT 'BOT'
        end;
#switch CL_TCP
#case REXTCPIP
        end;
      end;
      when (TCPRC=9) then do;
        /* wait, just a little bit longer ... */
        if (inXedit=1) then do;
          address XEDIT 'MSG waiting for' rsrv_name 'to respond...'
          address XEDIT 'REFRESH'
        end;
        'SLEEP 1 SEC'
      end;
      otherwise leave;
    end;/*select*/
#case RXSOCKET
      if (is_text=0) then leave;        /* binary, only one record...  */
                                        /* (is *THIS* true???????)     */
    end/*do forever */
#endswitch
  end;/*do forever*/
end;

if fm_opt='PUSH' then push('BOT');
if fm_opt='QUEUE' then queue('BOT');

#switch CL_TCP
#case REXTCPIP
'NUCXDROP REXTCPIP'
#case RXSOCKET
'NUCXDROP RXSOCKET'
#endswitch

if fm_opt='' then 'FINIS' fnm

exit(0);
```

4.4.2 prclin

```
/* ------------------------------------------------------------------ */
prclin: parse arg fnm, fm_opt, ty, line
select;
  when ty=1 then do;  /* directory ... */
    dty=substr(line,1,1)
    parse var line dtxt '05'x dsel '05'x dhst '05'x dprt '05'x nhit
    longline=0;
    if length(dtxt)>72 then longline=1;
    dtxt= substr(dtxt,2,72);
    if nhit/='' then dtxt=substr('('nhit')' dtxt,1,72);
    /* if length(dtxt)>72 then longline=1; */
    if longline/=0 then dtxt= dtxt'**';
    dtxt= substr(dtxt,1,75);
    select;
      /* initially defined types: - - - - - - - - - - - - - - - - - - - */
      when dty='0' then xdty='<F>'; /* normal file                     */
```

```
      when dty='1' then xdty='<D>'; /* normal directory              */
      when dty='2' then xdty='CSO'; /* CSO phonebook                 */
      when dty='7' then xdty='<S>'; /* search request: -> directory  */
      when dty='8' then xdty='<T>'; /* telnet session                */
      /* common extensions: - - - - - - - - - - - - - - - - - - - - - */
      when dty='T' then xdty='<T>'; /* 3270 telnet session           */
      when dty='w' then xdty='<S>'; /* search request: -> file       */
      /* panda extensions: - - - - - - - - - - - - - - - - - - - - - - */
      when dty='i' then xdty='***'; /* information entry             */
      /* local extensions: - - - - - - - - - - - - - - - - - - - - - - */
      when dty='B' then xdty='<B>'; /* BSS encoded                   */
      when dty='H' then xdty='>H<'; /* BESt encoded HyperText (HYX)  */
      otherwise xdty='>'dty'<';
    end;/*select*/
    if (dtxt/='' & dhst/='' & dprt/='') | xdty='***' then do;
      x= emmit(fnm, fm_opt, xdty dtxt dty dhst dprt dsel);
    end;
  end;
  when ty='2' then do;
    parse var line resp line2
    parse var resp resp1':'resp2':'
    if resp1='-200' then do;
      if resp2/=CSO_n then do;
        x= emmit(fnm, fm_opt, copies('-',64));
        CSO_n= resp2;
      end;
    end; else line2= line;
    x= emmit(fnm, fm_opt, line2);
  end;
  otherwise x= emmit(fnm, fm_opt, line);
end;/*select*/
return 0;
```

4.4.3 emmit

```
/* ------------------------------------------------------------------ */
emmit: parse arg fnm, fm_opt, line

select;
  when fm_opt='PUSH' | fm_opt='QUEUE' then do;
    do while length(line)>250;
      subline= substr(line,1,250);
      line= substr(line,251);
      if fm_opt='PUSH' then push('+'subline); else queue('+'subline);
    end;
    if fm_opt='PUSH' then push('-'line); else queue('-'line);
  end;
  when fm_opt='XEDIT' then do;
    if line='' then line=' ';
    address xedit 'INPUT' line;
  end;
  otherwise do;
    XL.0= 1;
    XL.1= line;
    'EXECIO 1 DISKW' fnm '(STEM XL.'
  end/*otherwise*/
end/*select*/

return 0;
```

4.4.4 report

```
/* ------------------------------------------------------------------ */
report: parse arg fnm, fm_opt, line
return emmit(fnm, fm_opt, '*ERROR*' line);
```

4.4.5 localfetch

```
#switch CL_USE_LOCALFETCH
#case YES
/* LOCAL ACCESS ===================================================== */
localfetch: parse arg fn ft fm, fm_opt, ty, sel':'xfrom':'xto
select;
  when sel='' then
    return localget(fn ft fm, fm_opt, ty, indexfile gopherdisk);
  when sel='SPEC-COOKIE' then do;
    'ERASE' fn 'SPEC'ft fm
    'COOKIE' fn 'SPEC'ft fm
    return localget(fn ft fm, fm_opt, ty, fn 'SPEC'ft fm);
  end;
  when left(sel,11)='SPEC-FINGER' then do;
    sel=substr(sel,13);
    'ERASE' fn 'SPEC'ft fm
    'FINGERXX' fn 'SPEC'ft fm '(0' sel
    return localget(fn ft fm, fm_opt, ty, fn 'SPEC'ft fm);
  end;
  otherwise do; /* try to find it from the file */
    'LISTFILE' selectorfile gopherdisk '(ALLOC LIFO NOHEAD'
    if rc/=0 then return -1;
    pull . . . . . xsiz .
    parse var sel xsfn'.'xsft'.'xsfm
    'FINIS' selectorfile
    do i=1 to xsiz;
      'EXECIO 1 DISKR' selectorfile '(LIFO'
      parse pull xfntm . . xsel
      if (xfntm/='*') then do;
        parse var xfntm xfn'.'xft'.'xfm
        if xft='*' & xfn=xsfn then do;
          x=localget(fn ft fm, fm_opt, ty, xfn xsft xfm, xfrom xto);
          'FINIS' selectorfile
          return x;
        end;
        if xsel=sel then do;
          x=localget(fn ft fm, fm_opt, ty, xfn xft xfm, xfrom xto);
          'FINIS' selectorfile
          return x;
        end;
      end;
    end;
    'FINIS' selectorfile
  end;
end;/*select*/
return -1;  /* selector not found */
#endswitch
```

4.4.6 localget

```
#switch CL_USE_LOCALFETCH
#case YES
```

```
/* ------------------------------------------------------------------- */
localget: parse arg fn ft fm, fm_opt, ty, xfn xft xfm, xfrom xto .
if xfm='' then xfm=gopherdisk;
xfm='*';   /* ##### */
q=queued();
'LISTFILE' xfn xft xfm '(ALLOC LIFO NOHEAD'
if rc/=0 then return -1;
q=queued()-q;
do i=1 to q;
  pull . . xfm . . ysiz .
end;
if xfrom=''|xfrom<0 then xfrom=0;
if xto=''|xto>ysiz then xto=ysiz;
'FINIS' xfn xft xfm
do i2=xfrom+1 to xto;
  'EXECIO 1 DISKR' xfn xft xfm i2 '(STEM XL.'
  x=prclin(fn ft fm, fm_opt, ty, XL.1);
end;
'FINIS' xfn xft xfm
return 0;
#endswitch
```

4.4.7 xconnstat

```
#switch CL_TCP
#case REXTCPIP
/* DEBUGGING ========================================================= */
xconnstat: parse arg gcn msg
say msg
connstat(gcn);
say 'open_time_wait='open_time_wait
say 'bytes_to_read='bytes_to_read
say 'unacked_bytes='unacked_bytes
say 'local_addr='local_addr
say 'local_port='local_port
say 'foreign_addr='foreign_addr
say 'foreign_port='foreign_port
say 'conn_stat='conn_stat
return 0;
#endswitch


#switch CL_TCP
#case RXSOCKET
```

4.4.8 EBCDIC to ASCII conversion

```
/* E2A EXEC simple ebcdic to ascii translate by JCA ------------------ */
E2A:
   return translate(arg(1), ,
     '000102030009007F0000000B0C0D0E0F'x||,
     '101112130000080018190000１C1D1E1F'x||,
     '00000000000A171B0000000000050607'x||,
     '0000160000000004000000001415001A'x||,
     '20000000000000000000002E3C282B7C'x||,
     '26000000000000000000021242A293B7E'x||,
     '2D2F000000000000000002C255F3E3F'x||,
     '005E00000000000000603A2340273D22'x||,
```

```
     '0061626364656666676869007B00000000'x||,
     '006A6B6C6D6E6F707172007D00000000'x||,
     '007E737475767778797A0000005B0000'x||,
     '000000000000000000000000005D0000'x||,
     '7B414243444546474849000000000000'x||,
     '7D4A4B4C4D4E4F505152000000000000'x||,
     '5C005354555657585950000000000000'x||,
     '30313233343536373839000000000000'x);
```

4.4.9 ASCII to EBCDIC conversion

```
/* A2E EXEC simple ascii to ebcdic translate by JCA ------------------ */
A2E:
   return translate(arg(1), ,
     '00010203372D2E2F1605250B0C0D0E0F'x||,
     '101112133C3D322618193F271C1D1E1F'x||,
     '405A7F7B5B6C507D4D5D5C4E6B604B61'x||,
     'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F'x||,
     '7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6'x||,
     'D7D8D9E2E3E4E5E6E7E8E9ADE0BD716D'x||,
     '79818283848586878889919293949596'x||,
     '979899A2A3A4A5A6A7A8A9C04FD05F07'x||,
     '00010203372D2E2F1605250B0C0D0E0F'x||,
     '101112133C3D322618193F271C1D1E1F'x||,
     '405A7F7B5B6C507D4D5D5C4E6B604B61'x||,
     'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F'x||,
     '7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6'x||,
     'D7D8D9E2E3E4E5E6E7E8E9ADE0BD716D'x||,
     '79818283848586878889919293949596'x||,
     '979899A2A3A4A5A6A7A8A98B4F9B5F07'x);



#endswitch
```

4.5 Module GOPHRING XEDIT

```
#erase GOPHRING XEDIT
#module GOPHRING XEDIT
/*********************************************************************/
/* File GOPHRING XEDIT  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME># */
/*                                                                   */
/* present XEDIT ring list on screen                                 */
/*                                                                   */
/* written:       1992-07-23 <Gerhard.Gonter@wu-wien.ac.at>          */
/* latest update: 1993-05-23 <>                                      */
/*********************************************************************/

'EXTRACT /RING/';
ringfile= 'GOPHTMP RING A'

outc= 0;
do i=2 to ring.0;
  parse var ring.i fni fti .
  if fni='GOPHTMP' then do;
    /* @@@ 'GLOBALXV SELECT GOPHER STACK GI1:'fti @@@ */
    address '' 'GLOBALV SELECT GOPHER STACK GI1:'fti
    parse pull seli
```

```
    outc= outc+1;
    XOUT.outc= substr(seli,1,80)||fti
  end;
end;
XOUT.0=outc;

'ERASE' ringfile
'EXECIO' outc 'DISKW' ringfile '(FINIS STEM XOUT.';
/* first: call the editor to get the right file */
'XEDIT' ringfile '(PROFILE GOPHER'
/* second: tell the editor to use the right profile mode */
'GOPHER RNG'
exit(0);
```

4.6 Module GOPHRNG3 XEDIT

```
#erase GOPHRNG3 XEDIT
#module GOPHRNG3 XEDIT
/*********************************************************************/
/* File GOPHRNG3 XEDIT  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME># */
/*                                                                   */
/* display selected item from the XEDIT ring                         */
/*                                                                   */
/* written:       1992-07-23 <Gerhard.Gonter@wu-wien.ac.at>          */
/* latest update: 1993-05-21 <>                                      */
/*********************************************************************/

/* get line of cursor */
'EXTRACT /CURSOR/FLSCREEN/FTYPE/';

if (CURSOR.3 < 0) then exit(0);
':'CURSOR.3
'EXTRACT /CURLINE/'
':'FLSCREEN.1

/* analyze item to be fetched */
lin= curline.3
lin= substr(lin,80);
'QUIT'
'XEDIT GOPHTMP' lin

exit(0);
```

4.7 Module GOPHSCRN XEDIT

```
#erase GOPHSCRN XEDIT
#module GOPHSCRN XEDIT
/*********************************************************************/
/* File GOPHSCRN XEDIT  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME>#  */
/*                                                                   */
/* screen handling: splitting and joining                            */
/*                                                                   */
/* written:     1992-07-23 <Gerhard.Gonter@wu-wien.ac.at>            */
/* last update: 1992-10-23 <>                                        */
/*********************************************************************/

parse arg subcmd .
```

```
upper subcmd;

'EXTRACT /SCREEN/'

x=words(screen.1)-1;
select;
  when subcmd='SPLIT' then 'COMMAND SET SCREEN' x+1;
  when subcmd='JOIN'  then 'COMMAND SET SCREEN' x-1;
  otherwise do; /* nothing -- just ignore that ... */ end;
end;
/* error checking?? plausibility test??? -- nah! XEDIT does that :)) */

exit(0);
```

4.8 Module GOPHSRCH XEDIT

```
#erase GOPHSRCH XEDIT
#module GOPHSRCH XEDIT
/*********************************************************************/
/* File GOPHSRCH XEDIT  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME># */
/*                                                                   */
/* Setup search screen                                               */
/* usually called by GOPHXFTC.XEDIT (*) search()                     */
/*                                                                   */
/* written:       1992-04-13 <Gerhard.Gonter@wu-wien.ac.at>          */
/* latest update: 1993-04-14 <>                                      */
/*********************************************************************/

'EXTRACT /LSCREEN/'
p0=lscreen.1-2;

parse pull SRrprt
parse pull SRrhst
parse pull SRrsel
parse pull SRdis
parse pull SRrty
parse pull SRcso_opts

'set tofeof off'
'set scale off'
'set prefix off'
'set reserved 1 nohigh',
'              ***** ***** Gopher Search Form ***** *****'
'set reserved 2 nohigh line 2'

'set reserved' p0 'nohigh PF3=Quit   PF5=Enter  (Enter=Enter too...)'
'set ctlchar % protect red high'
'set ctlchar $ noprotect blue high'
'set ctlchar " escape'

ph='"%'
nh='"$'

/* mask */
'set reserved 3 nohigh' SRdis
'set reserved 4 nohigh I will search on' SRrhst'('SRrprt') for'
z.5= ph'Search item:'
z.6= nh||copies(' ',76)||ph
```

```
if SRrsel/='' then do;
  'set reserved 7 nohigh and present the search string'
  'set reserved 8 nohigh' SRrsel
end;

/* 'EXTRACT /LSCREEN/' the same as above? (1993-04-14) */
do sline= 1 to lscreen.1
  if symbol('Z.SLINE')='VAR' then 'set reserved' sline 'nohigh' z.sline
end

/* readout data */
'cursor screen 6 2'
'read all tag'
'qquit'
```

4.9 Module GOPHXBMC XEDIT

```
#erase GOPHXBMC XEDIT
#module GOPHXBMC XEDIT
/*********************************************************************/
/* File GOPHXBMC XEDIT  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME># */
/*                                                                   */
/* save item pointed to by cursor into the bookmark file             */
/*                                                                   */
/* written:       1992-02-28 <Gerhard.Gonter@wu-wien.ac.at>          */
/* latest update: 1993-03-28 <>                                      */
/*********************************************************************/

/* get line of cursor */
'EXTRACT /CURSOR/FLSCREEN/'
if (CURSOR.3 < 0) then exit(0);
':'CURSOR.3
'EXTRACT /CURLINE/'
':'FLSCREEN.1

'GLOBALV SELECT GOPHER STACK BOOKMARK'
pull bookmark
if bookmark='' then bookmark='GOPHER BOOKMARK A';
'EXECIO 1 DISKW' bookmark '(FINIS STRING' CURLINE.3

exit(0);
```

4.10 Module GOPHXBMI XEDIT

```
#erase GOPHXBMI XEDIT
#module GOPHXBMI XEDIT
/*********************************************************************/
/* File GOPHXBMI XEDIT  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME>#  */
/*                                                                   */
/* save pointer to item currently displayed into the bookmark file   */
/*                                                                   */
/* written:     1992-02-28 <Gerhard.Gonter@wu-wien.ac.at>            */
/* last update: 1993-05-23 <>                                        */
/*********************************************************************/

'EXTRACT /FTYPE/';

/* identify the item being displayed */
```

```
/* @@@ 'GLOBALXV SELECT GOPHER STACK GI1:'FTYPE.1 @@@ */
address '' 'GLOBALV SELECT GOPHER STACK GI1:'FTYPE.1
parse pull gitem1
/* @@@ 'GLOBALXV SELECT GOPHER STACK GI2:'FTYPE.1 @@@ */
address '' 'GLOBALV SELECT GOPHER STACK GI2:'FTYPE.1
parse pull gitem2

/* find out where the bookmark is */
'GLOBALV SELECT GOPHER STACK BOOKMARK'
pull bookmark
if bookmark='' then bookmark='GOPHER BOOKMARK A';

/* save the bookmark entry */
'EXECIO 1 DISKW' bookmark '(FINIS STRING' gitem1||gitem2

exit(0);
```

4.11 Module GOPHXBMS XEDIT

```
#erase GOPHXBMS XEDIT
#module GOPHXBMS XEDIT
/*********************************************************************/
/* File GOPHXBMS XEDIT  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME>#  */
/*                                                                   */
/* display current bookmark file                                     */
/*                                                                   */
/* written:     1992-02-28 <Gerhard.Gonter@wu-wien.ac.at>            */
/* last update: 1993-05-26 <>                                        */
/*********************************************************************/

'GLOBALV SELECT GOPHER STACK BOOKMARK'
pull bookmark
if bookmark='' then bookmark='GOPHER BOOKMARK A';
parse var bookmark . fty .

'IDENTIFY (LIFO'
parse pull user . node .
uid='Bookmark list of <'user'@'node'>'

/* @@@ 'GLOBALXV SELECT GOPHER SETLP GHOST bookmark file:' uid bookmark @@@ */
/* @@@ 'GLOBALXV SELECT GOPHER SETLP GI1:'fty uid @@@ */
/* @@@ 'GLOBALXV SELECT GOPHER SETLP GI2:'fty bookmark @@@ */
address '' 'GLOBALV SELECT GOPHER SETL GHOST bookmark file:' uid bookmark
address '' 'GLOBALV SELECT GOPHER SETL GI1:'fty uid
address '' 'GLOBALV SELECT GOPHER SETL GI2:'fty bookmark

'XEDIT' bookmark '(PROFILE GOPHER'

exit(0);
```

4.12 Module GOPHXFTC XEDIT

```
#erase GOPHXFTC XEDIT
#module GOPHXFTC XEDIT
/*********************************************************************/
/* File GOPHXFTC XEDIT  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME>#  */
/*                                                                   */
/* Fetch item pointed to by the cursor and display it                */
```

```
/*                                                                    */
/* written:       1992-02-26 <Gerhard.Gonter@wu-wien.ac.at>           */
/*                1993-03-21 <> redesign/paraemter exchange via stack  */
/* latest update: 1993-05-26 <>                                       */
/**********************************************************************/

parfile='GOPHTMP FETCHPAR A';

/* get line of cursor */
'EXTRACT /CURSOR/FLSCREEN/FTYPE/';

if (CURSOR.3 < 0) then exit(0);
':'CURSOR.3
'EXTRACT /CURLINE/'
':'FLSCREEN.1

/* analyze item to be fetched */
lin= curline.3
dis= strip(substr(lin,1,78));
lin= substr(lin,80);
parse var lin rty rhst rprt rsel
rrty= rty;

select;
  when rty='0' | rty='1' | rty='2' | rty='7' | rty='B' | rty='w' then do;
    if rty='2' | rty='7' | rty='w' then do;
      if (search(rty,dis,rsel,rhst,rprt)/=0) then exit(0);
      if rty='7' then rty='1'; /* now treat it like a directory */

      if rty='w' then rty='0'; /* now treat it like a file */
    end;
    /* @@@ 'GLOBALXV SELECT GOPHER STACK TMPCNT' @@@ */
    address '' 'GLOBALV SELECT GOPHER STACK TMPCNT'
    pull tmpcnt .
    if tmpcnt='' then tmpcnt=1;
    'GLOBALV SELECT GOPHER SETL L'FTYPE.1 cursor.1 cursor.2
    /* @@@ 'GLOBALXV SELECT GOPHER SETLP TMPCNT' tmpcnt+1 @@@ */
    address '' 'GLOBALV SELECT GOPHER SETL TMPCNT' tmpcnt+1
    tmpfile='GOPHTMP TMP'tmpcnt 'A';
    fty='TMP'tmpcnt
/* @@@ 'GLOBALXV SELECT GOPHER SETLP GHOST' rhst':'rprt '<'rty'>' rsel @@@ */
/* @@@ 'GLOBALXV SELECT GOPHER SETLP GI1:'fty substr(curline.3,1,80); @@@ */
/* @@@ 'GLOBALXV SELECT GOPHER SETLP GI2:'fty substr(curline.3,81); @@@ */
address '' 'GLOBALV SELECT GOPHER SETL GHOST' rhst':'rprt '<'rty'>' rsel
address '' 'GLOBALV SELECT GOPHER SETL GI1:'fty substr(curline.3,1,80);
address '' 'GLOBALV SELECT GOPHER SETL GI2:'fty substr(curline.3,81);

    /* CSO integration here is ... ugly ...     */
    if rty='2' then do;
      XL.0=4;
      XL.1=rty;
      XL.2=rhst;
      XL.3=rprt;
      XL.4=rsel;
      if rty='2' then do;
        XL.0= 5;
        XL.4= 'query name='rsel;
        XL.5= 'quit';
      end;
```

```
    'ERASE' parfile
    'EXECIO' XL.0 'DISKW' parfile '(FINIS STEM XL.'
    'GOPHFTCH' tmpfile 'FILE' parfile
    'XEDIT' tmpfile '(PROFILE GOPHER'
    'ERASE' tmpfile
    exit(0);          /* brrr....              */
    /* or maybe it's not even that bad ...     */
    /* ... anyway, I keep an eye on this!      */
  end;

  queue('Host='rhst);
  queue('Port='rprt);
  queue('Type='rty);
  xrsel=rsel;
  do while length(xrsel)>240;
    queue('Path='substr(xrsel,1,240));
    xrsel= substr(xrsel,241);
  end;
  queue('Path='xrsel);
  queue('End');

  if rrty='7' then do;
    /* type 7 items go via file because I don't know how to   */
    /* hanle lines longer than 250 chars with xedit input.    */
    'GOPHFTCH' tmpfile 'PULL'
    'XEDIT' tmpfile '(PROFILE GOPHER'
    'ERASE' tmpfile
  end; else do;
   if 'all'='all' then do;
    if rty='1' then linewidth=250;
    else linewidth=80;
    'XEDIT' tmpfile '(WIDTH' linewidth 'PROFILE GOPHER'
    'RECFM V'
    'LRECL' linewidth
    'TRUNC' linewidth
    'SET LINEND OFF'
    ':1'
    'GOPHFTCH' tmpfile'(XEDIT PULL'
    ':1'
    'SET ALT 0 0'
   end;
   if 'this'='not used' then do;  /* move this to trash later on... */
    'GOPHFTCH' tmpfile'(QUEUE PULL'

    /* read the data from the stack */
    linecnt= 1;
    linewidth= 0;
    XLINE.1= '';
    q= queued();
    do q;
      parse pull line;
      xcode= substr(line,1,1);
      line= substr(line,2);
      select;
        when xcode='-' then do;
          XLINE.linecnt=XLINE.linecnt||line
          ll= length(XLINE.linecnt);
          linecnt= linecnt+1;
          XLINE.linecnt= '';
```

```
          if (ll>linewidth) then linewidth= ll;
        end;
        when xcode='+' then XLINE.linecnt= XLINE.linecnt||line;
        otherwise nop;
      end;
    end;
    linecnt= linecnt-1;

    'XEDIT' tmpfile '(WIDTH' linewidth 'PROFILE GOPHER'
    'ERASE' tmpfile

    /* get the data into the Xedit buffer */
    'RECFM V'
    'LRECL' linewidth
    'TRUNC' linewidth
    'SET LINEND OFF'
    ':1'
    do i=1 to linecnt;
      if XLINE.i='' then XLINE.i=' ';
      address xedit 'INPUT' XLINE.i;
    end;
    drop XLINE.;
    ':1'
   end;
  end;

end;
when rty='8' | rty='T' then do; /* TELNET or 3270 Session */
  'GLOBALV SELECT GOPHER STACK TELNET-'rty
  pull telnet_program
  if telnet_program='' then do;
    select;
      when rty='8' then telnet_program=#<CL_TELNET_8>#;
      when rty='T' then telnet_program=#<CL_TELNET_T>#;
      otherwise telnet_program='TELNET MODULE';
      /* Note: the otherwise path should never be taken anyway... */
    end/*selelct*/
  end;
  'STATE' telnet_program
  if rc/=0 then do;
    telnet_program='('telnet_program')'
    'COMMAND MSG sorry... you need TELNET'telnet_program 'to use this'
    exit(0);
  end;
  say '>>>> >>>> TELNET SESSION STARTING UP <<<< <<<<'
  'IDENTIFY (LIFO'
  pull me .
  if (rsel/='') then do;
    'TELL' me 'Enter' rsel 'on the login prompt'
    say 'Enter' rsel 'on the login prompt'
  end;
  if rprt<=0 then rprt='';
  word(telnet_program,1) rhst rprt;
end;
when rty='H' then do;
  'command msg sorry... cant''t handle this kind of HyperText yet'
  'command msg contact <Gerhard.Gonter@wu-wien.ac.at> for more info!'
end;
when rty='i' then do;
```

```
    /* nothing (panda info item) */
    'command msg That''s just info text; sorry...'
  end;
  when rty='3' then do;
    /* error return item */
    'command msg Hmmm...  this was just an error message; sorry...'
  end;
  otherwise do;
    'command msg sorry... cant''t handle this request currently'
  end;
end;

exit(0);

/* ------------------------------------------------------------------ */
search: parse arg SRrty,SRdis,SRrsel,SRrhst,SRrprt;
rv=0;
q=queued();
push 'name';           /* CSO options */
push SRrty
push SRdis
push SRrsel
push SRrhst
push SRrprt
'XEDIT GOPHER TMPSRCH A (PROFILE GOPHSRCH'
q=queued()-q;
do q;
  parse pull tag tline tcolumn ttext
  ttext=strip(ttext);
  if (tag='PFK' & tline='3') then return -1;
  if (tline=6) & (tcolumn=2) then do;
    if rsel/='' then rsel = rsel||'05'x||ttext; else rsel= ttext;
  end;
end;/*do q*/
/* if (rsel='') then return -1; ... empty search string *IS* allowed */
return rv;
```

4.13 Module GOPHXQIT XEDIT

```
#erase GOPHXQIT XEDIT
#module GOPHXQIT XEDIT
/********************************************************************/
/* File GOPHXQIT XEDIT  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME>#  */
/*                                                                  */
/* leave a file and restore some parameters                         */
/*                                                                  */
/* written:      1992-06-29 <Gerhard.Gonter@wu-wien.ac.at>          */
/* latest update: 1993-03-28 <>                                     */
/********************************************************************/

'QQUIT';
'COMMAND EXTRACT /FTYPE/';
if FTYPE.1/='FTYPE.1' then do;
  'GLOBALV SELECT GOPHER STACK L'FTYPE.1
  pull xl
  'COMMAND CURSOR SCREEN' xl
end;
```

```
exit(0);
```

4.14 Module GOPHXSVI XEDIT

```
#erase GOPHXSVI XEDIT
#module GOPHXSVI XEDIT
/*********************************************************************/
/* File GOPHXSVI XEDIT  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME>#  */
/*                                                                   */
/* save current item into file GOPHER SAVEFILE                       */
/*                                                                   */
/* written:       1992-02-28 <Gerhard.Gonter@wu-wien.ac.at>          */
/* latest update: 1993-05-23 <>                                      */
/*********************************************************************/

'EXTRACT /FLSCREEN/FTYPE/';

'GLOBALV SELECT GOPHER STACK SAVEFILE'
pull savefile
if savefile='' then savefile='GOPHER SAVEFILE A';

/* @@@ 'GLOBALXV SELECT GOPHER STACK GI1:'FTYPE.1 @@@ */
'GLOBALV SELECT GOPHER STACK GI1:'FTYPE.1
parse pull gitem1
/* @@@ 'GLOBALXV SELECT GOPHER STACK GI2:'FTYPE.1 @@@ */
'GLOBALV SELECT GOPHER STACK GI2:'FTYPE.1
parse pull gi2a gi2b gi2c gi2d

ds=date(sorted);
ds=substr(ds,1,4)'-'substr(ds,5,2)'-'substr(ds,7,2);

/* write header for saved item */
'EXECIO 1 DISKW' savefile '(STRING' copies('*',72);
'EXECIO 1 DISKW' savefile '(STRING saved GOPHER Item' ds time()
'EXECIO 1 DISKW' savefile '(STRING Name='gitem1
'EXECIO 1 DISKW' savefile '(STRING Type='gi2a
'EXECIO 1 DISKW' savefile '(STRING Host='gi2b
'EXECIO 1 DISKW' savefile '(STRING Port='gi2c
'EXECIO 1 DISKW' savefile '(STRING Path='gi2d
'EXECIO 1 DISKW' savefile '(FINIS STRING' copies('-',72);

/* write the item to be saved */
':1'
'PUT *' savefile
':'FLSCREEN.1

'command msg item saved to file' savefile
exit(0);
```

4.15 Module GOPHSTP EXEC

```
#erase GOPHSTP EXEC
#module GOPHSTP EXEC
/* ***************************************************************** */
/* File GOPHSTP.EXEC #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME># */
/*                                                                   */
/* Edit Gopher setup parameters using a mask                         */
/* Take also a look at the documentation: (*) User Setup             */
```

```
/*                                                                  */
/* written:       1993-05-20 <Gerhard.Gonter@wu-wien.ac.at>         */
/* latest update: 1993-05-31 <>                                     */
/* **************************************************************** */

'MAKEBUF'
group='GOPHER'

address '' 'GLOBALV SELECT' group 'STACK HOST PORT ITYPE SEL DNAME'
parse pull host .
parse pull port .
parse pull itype .
parse pull sel
parse pull dname

address '' 'GLOBALV SELECT' group 'STACK BOOKMARK SAVEFILE INFOLEVEL'
parse pull bookmark
parse pull savefile
parse pull infolevel .

address '' 'GLOBALV SELECT' group 'STACK TELNET-8 TELNET-T'
parse pull telnet_8
parse pull telnet_T

do forever;
  /* init variables */
  update= 1;

  q=queued();
  push host
  push port
  push itype
  push sel
  push dname
  push bookmark
  push savefile
  push infolevel
  push telnet_8
  push telnet_T

  'XEDIT GOPHER SETUP A (PROFILE GOPHSTP'
  q=queued()-q;

  do q;
    parse pull tag tline tcol ttxt;
    ttxt= strip(ttxt);

    if (tag='PFK' & tline='3') then do;
      done= 'YES';
      valid= 0;
      leave;
    end;

    if (tag='PFK' & tline='10') then do;
      dname= '';
      host= '';
      port= '';
      sel= '';
      itype= '';
```

```
    bookmark= '';
    savefile= '';
    telnet_8= '';
    telnet_t= '';
    infolevel= '';
    address '' 'GLOBALV SELECT' group 'SETLP DNAME' dname;
    address '' 'GLOBALV SELECT' group 'SETLP HOST' host;
    address '' 'GLOBALV SELECT' group 'SETLP PORT' port;
    address '' 'GLOBALV SELECT' group 'SETLP SEL' sel;
    address '' 'GLOBALV SELECT' group 'SETLP ITYPE' itype;
    address '' 'GLOBALV SELECT' group 'SETLP BOOKMARK' bookmark;
    address '' 'GLOBALV SELECT' group 'SETLP SAVEFILE' savefile;
    address '' 'GLOBALV SELECT' group 'SETLP TELNET-8' telnet_8;
    address '' 'GLOBALV SELECT' group 'SETLP TELNET-T' telnet_t;
    address '' 'GLOBALV SELECT' group 'SETLP INFOLEVEL' infolevel;
  end;

  if (tag='RES' & tline='4' & tcol='8') then do;
    dname= ttxt;
    address '' 'GLOBALV SELECT' group 'SETLP DNAME' dname;
  end;
  if (tag='RES' & tline='5' & tcol='8') then do;
    host= ttxt;
    address '' 'GLOBALV SELECT' group 'SETLP HOST' host;
  end;
  if (tag='RES' & tline='6' & tcol='8') then do;
    port= ttxt;
    address '' 'GLOBALV SELECT' group 'SETLP PORT' port;
  end;
  if (tag='RES' & tline='7' & tcol='8') then do;
    sel= ttxt;
    address '' 'GLOBALV SELECT' group 'SETLP SEL' sel;
  end;
  if (tag='RES' & tline='8' & tcol='8') then do;
    itype= ttxt;
    address '' 'GLOBALV SELECT' group 'SETLP ITYPE' itype;
  end;
  if (tag='RES' & tline='10' & tcol='18') then do;
    bookmark= ttxt;
    address '' 'GLOBALV SELECT' group 'SETLP BOOKMARK' bookmark;
  end;
  if (tag='RES' & tline='11' & tcol='18') then do;
    savefile= ttxt;
    address '' 'GLOBALV SELECT' group 'SETLP SAVEFILE' savefile;
  end;
  if (tag='RES' & tline='13' & tcol='30') then do;
    telnet_8= ttxt;
    address '' 'GLOBALV SELECT' group 'SETLP TELNET-8' telnet_8;
  end;
  if (tag='RES' & tline='14' & tcol='30') then do;
    telnet_t= ttxt;
    address '' 'GLOBALV SELECT' group 'SETLP TELNET-T' telnet_t;
  end;
  if (tag='RES' & tline='16' & tcol='14') then do;
    infolevel= ttxt;
    address '' 'GLOBALV SELECT' group 'SETLP INFOLEVEL' infolevel;
  end;

end/*do q*/;
```

```
  upper done;
  if (left(done,1)='J' | left(done,1)='Y') then leave;
end/*do forever*/;

'DROPBUF';
exit(0);
```

4.16 Module GOPHSTP XEDIT

```
#erase GOPHSTP XEDIT
#module GOPHSTP XEDIT
/* ****************************************************************** */
/* File GOPHSTP.XEDIT #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME># */
/*                                                                    */
/* Edit Gopher setup, parameter mask                                  */
/*                                                                    */
/* written:       1993-05-20 <Gerhard.Gonter@wu-wien.ac.at>           */
/* latest update: 1993-05-31 <>                                       */
/* ****************************************************************** */

'set tofeof off'
'set scale off'
'set prefix off'
'set reserved 1 nohigh',
'              ***** ***** GOPHER SETUP EDITOR ***** *****'
/* 'set reserved 2 nohigh line 2' */
/* 'set reserved 20 nohigh PF2=add   PF3=Quit   PF5=Enter (replace entry)'*/
'set ctlchar % protect blue high'
'set ctlchar $ noprotect red high'
'set ctlchar { noprotect yellow high'
'set ctlchar } noprotect green high'
'set ctlchar " escape'
'set pf4 ='

parse pull tn_T
parse pull tn_8
parse pull infolevel
parse pull savefnm
parse pull bmkfnm
parse pull dname
parse pull sel
parse pull itype
parse pull port
parse pull host

tn_T=          substr(tn_t,           1, 52);
tn_8=          substr(tn_8,           1, 52);
infolevel=     substr(infolevel,      1,  2);
savefnm=       substr(savefnm,        1, 20);
bmkfnm=        substr(bmkfnm,         1, 20);
dname=         substr(dname,          1, 74);
sel=           substr(sel,            1, 74);
itype=         substr(itype,          1,  1);
port=          substr(port,           1,  5);
host=          substr(host,           1, 74);

ph= '"%'
nh= '"$'
```

```
yh= '"{'
gh= '"}'

'EXTRACT /LSCREEN/';
last_line= lscreen.1-2;

/* mask */
z.3= ph'Default Coordinates in Gopher Space to start from:'
z.4= ph'Name='||nh||dname||ph
z.5= ph'Host='||nh||host||ph
z.6= ph'Port='||nh||port||ph
z.7= ph'Path='||nh||sel||ph
z.8= ph'Type='||nh||itype||ph

z.10= ph'Bookmark file: '||nh||bmkfnm||ph
z.11= ph'Save file:     '||nh||savefnm||ph
z.13= ph'Telnet program for Type=8: '||nh||tn_8||ph
z.14= ph'Telnet program for Type=T: '||nh||tn_t||ph
z.16= ph'Infolevel: '||nh||infolevel||ph||'0..nothing  5..normal  9..much'

z.last_line= 'PF:  2=......  3=quit  5=......  '||,
             '6=....... 7=..... 8=.... 9=.....  10=clear';

do zc= 1 to last_line;
   if symbol('Z.ZC') = 'VAR' then,
      'set reserved' zc 'nohigh' z.zc
end

/* readout data */
'cursor screen 4 8'
'read all tag'
'qquit'
```

4.17 Module XG EXEC

```
#erase XG EXEC
#module XG EXEC
/*********************************************************************/
/* File XG EXEC  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME>#        */
/*                                                                   */
/* display a file in GOPHER fashion                                  */
/*                                                                   */
/* written:     1992 02 29 <Gerhard.Gonter@wu-wien.ac.at>            */
/* last update: 1992 10 23 <>                                        */
/*********************************************************************/
parse arg fn ft fm .

'SET EMSG OFF'
'XEDIT' fn ft fm '(PROFILE GOPHER'
'SET EMSG ON'
exit(rc);



# endswitch
# this ends the block of fixes for the client modules
```

5 Utilities for Server Functions

5.1 Module COOKIE EXEC

```
#erase COOKIE EXEC
#module COOKIE EXEC
/* ------------------------------------------------------------------ */
/* File COOKIE EXEC  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME>#     */
/*                                                                    */
/* a COOKIE (or FORTUNE) message generator                            */
/*                                                                    */
/* written:     1992-05-08 <Gerhard.Gonter@wu-wien.ac.at>             */
/* last update: 1992-10-23 <>                                         */
/* ------------------------------------------------------------------ */
parse upper arg fnm '('option

'MAKEBUF';
readdisk='*';
fortune_txt='FORTUNE TXT' readdisk
fortune_idx='FORTUNE IDX' readdisk

'LISTFILE' fortune_idx '(ALLOC LIFO NOHEAD'
if rc/= 0 then do;
  push 'there is no fortune cookie for you!'
  q=1;
end; else do;
  pull . . . . . fortune_size .
  r=random(1,fortune_size);
  'EXECIO 1 DISKR' fortune_idx r '(FINIS LIFO'
  pull from to
  size=to-from+1;
  /* say 'r='r 'from='from 'to='to 'size='size */

  q=queued();
  'EXECIO' size 'DISKR' fortune_txt from '(FINIS FIFO'
  q=queued()-q;
end;

if fnm='' then do;
  do q;
    parse pull line
    say line
  end;
end; else; do;
  'EXECIO' q 'DISKW' fnm '(FINIS'
end;

'DROPBUF'
exit(0);
```

5.2 Module FINGER EXEC

```
#erase FINGER EXEC
#module FINGER EXEC
/* ------------------------------------------------------------------ */
/* FINGER EXEC                                                        */
/*                                                                    */
/* Interface to contact RUI (remote user information) servers on      */
```

```
/* remote hosts or to get user information about local users.         */
/* For local user information, the program FINGERXX.EXEC is called.    */
/*                                                                     */
/* written:     1992 05 08: <Gerhard.Gonter@wu-wien.ac.at>             */
/* last update: 1992 06 19: ditto                                      */
/* ------------------------------------------------------------------- */
parse arg req

last_at=0;
do forever;
  i=index(req,'@',last_at+1);
  if (i=0) then leave;
  last_at=i;
end;

if last_at>0 then do;
  usr=substr(req,1,last_at-1);
  node=substr(req,last_at+1);
end; else do;
  usr=req;
  node='';
end;

localnode.1='awiwuw11.wu-wien.ac.at'

if node='' then do;
  say '['localnode.1']'
  'FINGERXX (0' usr
  exit(0);
end;

say '['node']'

remote=node;

'REXTCPIP'
if verify(node,'0123456789.') /= 0 then do;
  remote=GetIPAddr(node);
  if remote='' then do;
    say 'unknown host:' node
    'NUCXDROP REXTCPIP'
    exit(0);
  end;
end;

cn=TCPOPEN(remote,79,15);
if TCPRC=0 then do;
  tcpsend(cn,usr);
  connstat(cn);
  do forever;
    if conn_stat/='Connected' & conn_stat/='Sending only' then leave;
    tcpreceive(cn,'WAIT');
    if TCPRC/= 0 & TCPRC /= 18 then leave;
    do i=1 to tcpline.0;
      say TCPLINE.i
    end;
    connstat(cn);
  end;
end;
```

```
'NUCXDROP REXTCPIP'
exit(0);
```

5.3 Module FINGERXX EXEC

```
#erase FINGERXX EXEC
#module FINGERXX EXEC
```

5.3.1 main

```
/* ------------------------------------------------------------------ */
/* File FINGERXX EXEC  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME># */
/*                                                                    */
/* output preparation for RUI (remote user information)              */
/*   this program is normally called by GOPHERDD.EXEC to answer       */
/*   FINGER queries or by FINGER.EXEC for local user information.     */
/*                                                                    */
/* written:       1992-05-08 <Gerhard.Gonter@wu-wien.ac.at>           */
/* latest update: 1993-03-28 <>                                       */
/* ------------------------------------------------------------------ */
parse arg fnm '('qty qtxt

address COMMAND;

phonebook='PHONEBK'   /* NAMES file containing phone book information  */
localnames='LOCAL'    /* NAMES file containng info about local users   */
terminals='TERMINALS' /* NAMES file containng info about terminals     */
localhost='awiwuw11.wu-wien.ac.at'
localport='70';
selpfx='SPEC-FINGER';
verbose=0;

if fnm/='' & words(fnm)/=3 then do;
  say 'invalid filename' fnm
  exit(1);
end;

'MAKEBUF'
if fnm/='' then do;
  'STATE' fnm
  if rc=0 then 'ERASE' fnm
end;

tfields=0;
if index(qtxt,'@')>0 then do;
  disp 'Finger forwarding service denied'
  signal OVER;
end;
if substr(qtxt,1,2)='/W' then do;
  verbose=1;
  qtxt=substr(qtxt,3);
end;

uqtxt=qtxt;
upper uqtxt;

select;
  when substr(qtxt,1,1)='#' then x=chknames(1 substr(qtxt,2));
```

```
  when words(qtxt)=0 | qtxt='NAMES' then do;
    x=show_all_users();
  end;
  when words(qtxt)=1 & uqtxt='COOKIE' then do;
    'COOKIE' fnm
  end;
  when words(qtxt)=1 then do;
    x=chkusr(qtxt);
    x=chknames(2 qtxt);
  end;
  when words(qtxt)>1 & word(qtxt,1)='query' then do; /* CSO */
    parse var qtxt . qty'='qst
    qst=substr(qst,2,length(qst)-2);
    x=chkusr(qst);
    x=chknames(2 qst);
  end;
  otherwise do;
    x=chknames(3 qtxt);
  end;
end;

if tfields>0 then queue('### PHONEBOOK ###');
do i=1 to tfields;
  queue(fieldname.i'='fieldvalue.i)
end;

/********
if (verbose) then do;
  queue 'no extra verbose information available...'
end;
********/

OVER:
if fnm/='' then 'EXECIO' queued() 'DISKW' fnm '(FINIS'
else do queued();
  parse pull l
  say l
end;
'DROPBUF'
exit(0);
```

5.3.2 chkusr

```
/* ------------------------------------------------------------------- */
chkusr: parse upper arg usr
upper usr
select;
  when qty=1 then do;
    'EXECIO * CP (FIFO STRING QUERY' usr
    parse pull ulog
    push '0Finger:' usr||'05'x||selpfx usr||localhost||localport
  end;
  otherwise do;
    x=show_header();
    x=show_usr(usr 0);
    'STATE FINGERID' usr
    if rc=0 then do;
       queue('### ID-FILE ###');
      'EXECIO * DISKR FINGERID' usr '* (FIFO FINIS';
```

```
    end;
  end;
end;
return 0;
```

5.3.3 chknames

```
/* ------------------------------------------------------------------ */
chknames: parse arg mode str
cnt=0;
qq=queued();
select;
  when mode=1 then do;
    'NAMEFIND :phone' str ':nick :cnt :oid (LIFO * FILE' phonebook
    if rc/=0 then return -1;
  end;
  when mode=2 then do;
    'NAMEFIND :fname' str ':nick :cnt :oid (LIFO * FILE' phonebook
    if rc/=0 then return -1;
  end;
  when mode=3 then do;
    xstr=translate(str,'_',' ');
    'NAMEFIND :name' xstr ':nick :cnt :oid (LIFO * FILE' phonebook
    if rc/=0 then do;
      str=word(str,2) word(str,1);
      xstr=translate(str,'_',' ');
      'NAMEFIND :name' xstr ':nick :cnt :oid (LIFO * FILE' phonebook
      if rc/=0 then return -1;
    end;
  end;
  otherwise return 0;
end;

qq=queued()-qq;
tfields=1;
qq=qq/2;
do ii=1 to qq;
  parse pull fields
  parse pull nick
  xfields='';
  xtfields=tfields;
  do while fields/='';
    parse var fields field fields
    fieldname.tfields=field;
    tfields=tfields+1;
    xfields=':'field xfields
  end;
  q=queued();
  'NAMEFIND :nick' nick xfields '(LIFO * FILE' phonebook
  q=queued()-q;
  do i=xtfields to tfields-1;
    parse pull fieldvalue.i
    fieldvalue.i=translate(fieldvalue.i,' ','_');
  end;
end;
tfields=tfields-1;

return cnt;
```

5.3.4 disp

```
/* ------------------------------------------------------------------- */
disp: parse arg str
if fnm/='' then queue str;
else say str;
return 0;
```

5.3.5 show_header

```
/* ------------------------------------------------------------------- */
show_header: parse arg .
x=disp('Login    ' substr('Name',1,24) 'Terminal Phone/Location')
return 0;
```

5.3.6 show_usr

```
/* ------------------------------------------------------------------- */
show_usr: parse arg xusr mode .
'EXECIO * CP (LIFO STR QUERY' xusr
parse pull login_stat
if (login_stat='DMKCQG045E' xusr 'not logged on') |,
   (login_stat='HCPCQV003E Invalid option -' xusr) then do;
  terminal='offline'
end; else do;
  parse var login_stat . . terminal .
end;
if ((terminal='DSC'|terminal='offline') & mode=1) then return 0;

names_stat='???'
phone='';
place='';

q=queued();
'NAMEFIND :nick' terminal ':location (LIFO * FILE' terminals
q=queued()-q;
do q;
  parse pull x
  if place='' then place=x; else place= place x
end;

q=queued();
'NAMEFIND :nick' xusr ':name (LIFO * FILE' localnames
q=queued()-q;
if q then do;
  parse pull names_stat
  names_stat=translate(names_stat,' ','_');
  q=queued();
  'NAMEFIND :fname' word(names_stat,1) ':phone (LIFO * FILE' phonebook
  'NAMEFIND :fname' word(names_stat,words(names_stat)),
     ':phone (LIFO * FILE' phonebook
  q=queued()-q;
  phone='';
  do q;
    parse pull x
    if phone='' then phone=x; else phone=phone',' x
  end;
end;
```

```
locph=phone;
if locph='' then locph= place; else do;
  if place/='' then locph= locph';' place;
end;

queue substr(xusr,1,8) substr(names_stat,1,24),
      substr(terminal,1,8) locph
return 0;
```

5.3.7 show_all_users

```
/* ------------------------------------------------------------------ */
show_all_users: parse arg xusr
'EXECIO * CP (STEM QN. STRING QUERY NAMES'
x=show_header();
do i=1 to QN.0;
  parse var QN.i u.1 .',' u.2 .',' u.3 .',' u.4 .
  do j=1 to 4;
    if u.j/='' then x=show_usr(substr(u.j,1,8) 1);
  end;
end;
return 0;



/*********************************************************************/
/* REACCESS attached disks                                           */
/* latest update: 1993-05-03                                         */
/*********************************************************************/
parse upper arg fms
'MAKEBUF';
'EXECIO 1 CP (LIFO STR Q SET'
pull . . . . a.5 a.6 a.7 a.8 .
do i = 5 to 7
   if a.i = 'EMSG' then do
      j=i+1;
      emsg = a.j
      if emsg='IUCV,' then emsg='IUCV'
   end
end
'set emsg off'
q=queued();
'QUERY DISK (FIFO'
q=queued()-q-1;
pull .
reaccessed='';
do i = 1 to q
   pull . CUU MODE STAT .
   if stat='R/O' & mode /= 'S' then do
      'access' CUU MODE
      reaccessed = reaccessed CUU
   end
end
'set emsg' emsg
'DROPBUF';
'SUBCOM XEDIT'
if rc=0 then do;
  address XEDIT 'MSG disks' reaccessed 'reaccessed'
  address XEDIT 'REFRESH'
```

```
end; else say 'disks' reaccessed 'reaccessed'
exit(0);
```

6 Administration Support

6.1 Module EDINDEX EXEC

```
#erase EDINDEX EXEC
#module EDINDEX EXEC
/* *************************************************************** */
/* File EDINDEX.EXEC #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME># */
/*                                                                 */
/* Edit Gopher INDEX files using a mask                            */
/*                                                                 */
/* written:       1992-06-14 <Gerhard.Gonter@wu-wien.ac.at>        */
/* latest update: 1993-04-30 <>                                    */
/* *************************************************************** */
parse arg fn ft fm num '('opts

if fn='='|fn='' then fn='GOPHERD';
if ft='='|ft='' then ft='INDEX';
if fm='='|fm='' then fm='A';
if num='' then num=1;

done='';
valid=0;
tab='05'x;
GINDcnt=0;
SELcnt=0;
selfile='GOPHERD SELECTOR A';

'MAKEBUF';

'LISTFILE' fn ft fm '(NOH ALLOC LIFO'
if rc/=0 then do;
  GIND.0= 1;
  GIND.1= '  '||tab||tab||tab;
  GINDcnt= 1;
end; else do;
  pull . . . . . GINDcnt .
  'EXECIO * DISKR' fn ft fm '(FINIS STEM GIND.'
  if (num > GINDcnt) then num=1;
end;

/* items type=i should have 3 tabs at the end should contain at least  */
/* least a blank as the information entry ... (for poor unix boxes)    */
do i=1 to GIND.0;
  if substr(GIND.i,1,1)='i' then do;
    parse var GIND.i dtxt '05'x dsel '05'x dhst '05'x dprt '05'x .
    dtyp= substr(dtxt,1,1);
    dtxt= substr(dtxt,2);
    if dtxt='' then dtxt=' ';
    GIND.i=dtyp||dtxt||tab||tab||tab;
  end;
end;

'LISTFILE' selfile '(NOH ALLOC LIFO'
if rc=0 then do;
```

```
  pull . . . . . SELcnt .
  'EXECIO * DISKR' selfile '(FINIS STEM SEL.'
end;

do forever;
  /* init variables */
  parse var GIND.num dtxt '05'x dsel '05'x dhst '05'x dprt '05'x .
  dtyp= substr(dtxt,1,1);
  dtxt= substr(dtxt,2);
  update= 1;

  q=queued();
  push fn ft fm
  push GINDcnt
  push num
  push dtyp
  push dtxt
  push dsel
  push dhst
  push dprt

  'XEDIT EDINDEX TEMP A (PROFILE EDINDEX'
  q=queued()-q;

  snum=num;
  do q;
    parse pull tag tline tcol ttxt;
    ttxt= strip(ttxt);
    if (tag='PFK' & tline='2') then do;  /* append */
      GINDcnt= GINDcnt+1;
      num= GINDcnt;
      snum= GINDcnt;
    end;
    if (tag='PFK' & tline='3') then do;
      done= 'YES';
      valid= 0;
      leave;
    end;

    if (tag='PFK' & tline='5') then do;
      x= entry_insert(num);
      GIND.num=' ';
      update= 0;
    end;
    if (tag='PFK' & tline='6') then do;
      x= entry_delete(num);
      if num>GINDcnt then num=GINDcnt;
      update= 0;
    end;
    if (tag='PFK' & tline='7' & num > 1) then num=num-1;
    if (tag='PFK' & tline='8' & num < GINDcnt) then num=num+1;

    if (tag='PFK' & tline='9') then do;
      GIND.num=' ';
      update= 0;
    end;
    if (tag='PFK' & tline='10') then do;
      dprt= 70;
      dtyp= 0;
```

```
    dsel= ' ';
  end;

  if (tline =  5) & (tcol =  2) then dtxt= ttxt;
  if (tline =  8) & (tcol =  2) then dsel= ttxt;
  if (tline = 10) & (tcol =  2) then dhst= ttxt;
  if (tline = 12) & (tcol =  8) then dtyp= ttxt;
  if (tline = 12) & (tcol = 20) then dprt= ttxt;

 end/*do q*/;

 if update=1 then do;
  select;
    when dtyp='i' then do;
      if dtxt='' then dtxt=' ';
      GIND.snum =dtyp||dtxt||tab||tab||tab;
    end;
    otherwise GIND.snum=dtyp||dtxt||tab||dsel||tab||dhst||tab||dprt
  end/*select*/
 end;

 upper done;
 if (left(done,1)='J' | left(done,1)='Y') then leave;
end/*do forever*/;

'ERASE' fn ft fm
'EXECIO' GINDcnt 'DISKW' fn ft fm '(FINIS STEM GIND.'
'DROPBUF';

exit(0);

/* ------------------------------------------------------------------- */
entry_delete: parse arg n;

if GINDcnt<=1 then return -1;

do i=n to GINDcnt-1;
  j= i+1;
  GIND.i= GIND.j;
end;
GINDcnt= GINDcnt-1;
GIND.0= GINDcnt;

return 0;

/* ------------------------------------------------------------------- */
entry_insert: parse arg n;

do i=GINDcnt to n by -1;
  j= i+1;
  GIND.j= GIND.i;
end;
GINDcnt= GINDcnt+1;
GIND.0= GINDcnt;

return 0;
```

6.2 Module EDINDEX XEDIT

```
#erase EDINDEX XEDIT
#module EDINDEX XEDIT
/**********************************************************************/
/* File EDINDEX.XEDIT  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME># */
/*                                                                    */
/* Mask for editing Gopher INDEX files                                */
/*                                                                    */
/* written:       1992-08-08 <Gerhard.Gonter@wu-wien.ac.at>           */
/* latest update: 1993-04-30 <>                                       */
/**********************************************************************/

'set tofeof off'
'set scale off'
'set prefix off'
'set reserved 1 nohigh',
'              ***** ***** GOPHER INDEX EDITOR ***** *****'
/* 'set reserved 2 nohigh line 2' */
/* 'set reserved 20 nohigh PF2=add   PF3=Quit   PF5=Enter (replace entry)'*/
'set ctlchar % protect blue high'
'set ctlchar $ noprotect red high'
'set ctlchar { noprotect yellow high'
'set ctlchar } noprotect green high'
'set ctlchar " escape'
'set pf4 ='

parse pull dprt
parse pull dhst
parse pull dsel
parse pull dtxt
parse pull dtyp
parse pull num
parse pull totnum
parse pull fnm

dtxt=substr(dtxt,1,70);
dsel=substr(dsel,1,78);
dhst=substr(dhst,1,78);
dprt=substr(dprt,1,10);
ph = '"%'
nh = '"$'
yh = '"{'
gh = '"}'

/* mask */
z.3 = gh'File'yh fnm  gh'record'yh num gh'of'yh totnum||gh
z.4 = ph'Name:    '
do zc=1 to 7;
  z.4=z.4||zc||'         ';
end;
z.5 = nh||dtxt||ph
z.6 = ph||substr(copies('1234567890',8),1,78);
z.7 = ph'Path:'
z.8 = nh||dsel||ph
z.9 = ph'Host:'
z.10 = nh||dhst||ph

z.12 = ph'Type:'nh||dtyp||ph'    Port:'nh||dprt||ph
```

```
z.14 = 'important types:   0..normal file   1..directory   i..information'
z.15 = '7..search and return directory   w..search and return file'
z.16 = '9..telnet   T..3270 telnet'
z.17 = 'see also: * table of supported item types in VieGOPHER docu.'

'EXTRACT /LSCREEN/';
last_line= lscreen.1-2;

z.last_line= 'PF:  2=append  3=quit  5=insert  '||,
             '6=delete  7=back  8=next  9=clear  10=default';

do zc = 1 to last_line;
   if symbol('Z.ZC') = 'VAR' then,
      'set reserved' zc 'nohigh' z.zc
end

/* readout data */
'cursor screen 5 2'
'read all tag'
'qquit'
```

6.3 Module GOPHCHRL EXEC

```
#erase GOPHCHRL EXEC
#module GOPHCHRL EXEC
/*********************************************************************/
/* File GOPHCHRL EXEC  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME>#   */
/*                                                                   */
/* Check if there are interesting items in the GOPHER SERVICE VM's   */
/* virtual reader.                                                   */
/*                                                                   */
/* written:     1992-07-02 <Gerhard.Gonter@wu-wien.ac.at>            */
/* last update: 1992-10-23 <>                                        */
/*********************************************************************/
'EXECIO * CP (STEM R. STR Q TIME'
parse var R.1 . . . . . xtime
monat=delstr(xtime,3);
tag=delstr(xtime,6);
tag=delstr(tag,1,3);
if monat='10' then monat='0A'
if monat='11' then monat='0B'
if monat='11' then monat='0C'
monat=delstr(monat,1,1);

upd=0;

'EXECIO * CP (STEM R. STR QUERY READER ALL';
DO I = 2 TO R.0
  PARSE UPPER VAR R.I ORGID FIL CL1 CL2 RECS . . DAT TIM NAM TYP DIST
say ORGID FIL CL1 CL2 RECS . . DAT TIM NAM TYP DIST
  select;
    when NAM='FINGERID' then do;
      if TYP=ORGID then do;
        upd=1;
        'RECEIVE' FIL '(REPL';
      end;
    end;
    when ORGID='MAILER' & CL1='M' & CL2='PUN' then do;
```

```
say 'Mail|'
      'GOPHSTOR RM' FIL 'CHECK REPLY' ORGID NAM '.'
say 'RC='rc
    end;
    otherwise do; /* ignore it */ end;
  end/*select*/;
end;

if upd=1 then 'UPDINFO';
```

6.4 Module GOPHSTOR EXEC

```
#erase GOPHSTOR EXEC
#module GOPHSTOR EXEC
```

6.4.1 main

```
/***********************************************************************/
/* File GOPHSTOR EXEC  #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME># */
/*                                                                     */
/* **** THIS MODULE IS UNFINISHED ****                                 */
/*                                                                     */
/* Process a file or a spool file and store it if it's ok.             */
/*                                                                     */
/* written:      1992-07-03 <Gerhard.Gonter@wu-wien.ac.at>             */
/* latest update: 1993-04-30 <>                                        */
/***********************************************************************/
parse upper arg ty fil auth reply orgid sender emaddr .
say 'GOPHSTOR('ty fil auth reply orgid sender emaddr')'

selectorfile='GOPHERD SELECTOR';
gopherdisk='A';
rwdisk='A';
tmpnb='GOPHSTOR';
data_file='GOPHSTOR DATA' rwdisk;
owner_names='OWNER'
upd=0;

x= init_params();

'MAKEBUF'
select;
  when ty='RM' then do;
    'ERASE' tmpnb 'NOTEBOOK' rwdisk
    'RECEIVE' FIL '(NOTEBOOK' tmpnb
    x=proc_file('MAIL' tmpnb 'NOTEBOOK' rwdisk);
    if x=-1 then signal OVER;
  end;
  otherwise do; /* nothing */ end;
end/*select*/;

OVER:
'DROPBUF';
return upd;
```

6.4.2 init_params

```
/* ------------------------------------------------------------------- */
```

```
init_params:

/* Origin */
From= '';
Date= '';
Pass= '';        /* ... but this is not found in the header */
Subject= '';
Message_id= '';

/* information about the item to be stored; */
IDir= '';
IName= '';
IType= '';
IPath= '';
IHost= '';
IPort= '';
return 0;
```

6.4.3 proc_file

```
/* ---------------------------------------------------------------------- */
proc_file: parse arg mode fnm
say 'proc_file: mode='mode   'fnm='fnm

data_written=0;

q=queued();
'LISTFILE' fnm '(LIFO NOHEAD ALLOC'
if rc/=0 then return -1;
q=queued()-q;

do q;
  parse pull . . . . . fsiz .
end;

do fsiz;
  'EXECIO 1 DISKR' fnm '(STEM L.'
  lin=L.1
  select;
    when mode='MAIL' then do;
      select;
        when lin=''|lin=' ' then mode='FILE';
                        /*12345678901234567890*/
        when left(lin,6)='From: ' then From=strip(substr(lin,7));
        when left(lin,6)='Date: ' then Date=strip(substr(lin,7));
        when left(lin,9)='Subject: ' then Subject=strip(substr(lin,10));
        otherwise do; /* nothing */ end;
      end/*select*/;
    end;
    when mode='FILE' then do;
      select;
                        /*12345678901234567890*/
        when left(lin,4)='Dir=' then IDir=strip(substr(lin,5));
        when left(lin,5)='Name=' then IName=strip(substr(lin,6));
        when left(lin,5)='Type=' then IType=strip(substr(lin,6));
        when left(lin,5)='Path=' then IPath=strip(substr(lin,6));
        when left(lin,5)='Host=' then IHost=strip(substr(lin,6));
        when left(lin,5)='Port=' then IPort=strip(substr(lin,6));
        when left(lin,5)='Pass=' then  Pass=strip(substr(lin,6));
```

```
        when left(lin,5)='Data=' then do;
          mode='DATA';
          'ERASE' data_file
        end;
        otherwise do; /* nothing */ end;
      end/*select*/;
    end;
    when mode='DATA' then do;
      L.0=1;
      L.1=lin;
      data_written=1;
      'EXECIO 1 DISKW' data_file '(STEM L.'
    end;
    otherwise return -1; /* unknown processing mode ... */
  end/*select*/;
end;

if data_written then 'FINIS' data_file;

if IDir=''|IName=''|IType='' then return -1;

/* check, if user is authorized to write to the directory specified */
parse value get_filename(IDir)  with IDir_own  IDir_fnm
IDir_ok=check_owner(IDir_own,  Pass, From);
say IDir '->' IDir_own IDir_fnm IDir_ok
if IDir_ok/=1 then return -1;

/* check, if the file specified is ok or whatever */
parse value get_filename(IName) with IName_own IName_fnm
if IName_own='' then do;
  /* no owner yet ... */
end; else do; /* file has already an owner */
  IName_ok=check_owner(IName_own, Pass, From);
  say IName '->' IName_own IName_fnm IName_ok
  if INamoe_ok/=1 then return -1;
end;

return 0;
```

6.4.4 check_owner

```
/* ----------------------------------------------------------------- */
check_owner: parse arg own, pass, from
if own='*' then return 1;
'NAMEFIND :uid' word(from,1) ':pass :fac (LIFO * FILE' owner_names
parse pull ffac
parse pull fpass
if ffac=own & fpass=pass then return 1;
return 0;
```

6.4.5 get_filename

```
/* ----------------------------------------------------------------- */
get_filename: parse arg sel
parse value _get_filename(sel) with xown xfn xft xfm .
if xown='' then return '';
say xown fn xft xfm
/********************
```

```
q=queued();
'LISTFILE' xfn xft '* (ALLOC LIFO NOHEAD'
if rc/=0 then return '';
q=queued()-q;
pull . . xfm . . xsiz .
*****************/
return xown xfn xft gopherdisk;
```

6.4.6 _get_filename

```
/* ------------------------------------------------------------------- */
_get_filename: parse arg sel

'LISTFILE' selectorfile gopherdisk '(ALLOC LIFO NOHEAD'
if rc/=0 then return -1;
pull . . . . . xsiz .
parse var sel xsfn'.'xsft'.'xsfm
'FINIS' selectorfile
do i=1 to xsiz;
  'EXECIO 1 DISKR' selectorfile '(LIFO'
  parse pull xfntm xown . xsel
  if (xfntm/='*') then do;
    parse var xfntm xfn'.'xft'.'xfm
    if xft='*' & xfn=xsfn then do;
      'FINIS' selectorfile
      return xown xfn xsft xfm;
    end;
    if xsel=sel then do;
      'FINIS' selectorfile
      return xown xfn xft xfm;
    end;
  end;
end;
'FINIS' selectorfile
return '';  /* selector not found */
```

6.5 Module GOPHSCAN EXEC

```
#erase GOPHSCAN EXEC
#module GOPHSCAN EXEC
```

6.5.1 main

```
/* ------------------------------------------------------------------- */
/* File GOPHSCAN.EXEC #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME># */
/*                                                                     */
/* Scan Gopher menus from a (list of) given servers                    */
/*                                                                     */
/* written:       1993-05-05 <Gerhard.Gonter@wu-wien.ac.at>            */
/* latest update: 1993-05-26 <>                                        */
/* ------------------------------------------------------------------- */
parse arg servers'('opts

iniload='GOPHSCAN INITIAL A'    /* not used right now ...              */
tmpfile='TMP TMP A'             /* file to store menu from server      */
                                /* while examining the data there      */
outfile='GOPHSCAN OUT A';       /* final output of the scanner, all    */
                                /* items from all servers              */
```

```
def_relax= '1 sec';             /* delay before two scans           */
relax= '';                      /* next delay before scan           */

                                /* host on watch list               */
sc_num= 0;                      /* number of servers to scan        */
sc_host.='';                    /* servers to scan                  */
sc_port.='';                    /* ports where server lives on      */

                                /* things left to be done           */
t2d_num= 0;                     /* total number of dirs on work list  */
t2d_host.='';
t2d_port.='';
t2d_path.='';
t2d.=0;                         /* flag: menu must still be scanned   */

                                /* a little bit of statistics       */
dirs_scanned= 0;                /* number of directories scanned    */
dirs_ok= 0;                     /* number of directories received ok  */
lines_scanned= 0;               /* number of lines in directories   */

'ERASE' outfile;                /* create output file from the start  */

do while servers/='';
  parse var servers server servers
  parse var server host':'port
  if port='' then port= 70;
  x= add_host(host, port);
  x= add_item(host, port, '');
end;

upper opts;
do while opts/='';
  parse var opts opt opts
  if left(opt,6)='LIST=' then do;
    'EXECIO * DISKR' translate(substr(opt,7),' ','.') '(STEM LI.'
    do i=1 to LI.0;
      parse var host':'port
      if port='' then port= 70;
      x= add_host(host, port);
      x= add_item(host, port);
    end;
  end;
end;

if sc_num=0 then do;
  say 'please specify the server(s) that should be scanned:';
  say 'usage: GOPHSCAN host[:port]+';
  exit(0);
end;

thing= 1;
do forever;
  if thing > t2d_num then leave;
  if t2d.thing=1 then do;
    t2d.thing= get_item(t2d_host.thing, t2d_port.thing, t2d_path.thing);
  end;
  thing= thing+1;
end;
```

```
'ERASE' tmpfile;
say dirs_scanned 'directories scanned,' dirs_ok 'were ok.'
say lines_scanned 'lines written to' outfile'.'
exit(0);
```

6.5.2 add_host

```
/* ------------------------------------------------------------------ */
add_host: parse arg host, port
sc_num= sc_num+1;
sc_host.sc_num= host;
sc_port.sc_num= port;
return 0;
```

6.5.3 add_item

```
/* ------------------------------------------------------------------ */
add_item: parse arg host, port, path
say 'add_item: host='host 'port='port 'path='path
t2d_num= t2d_num+1;
t2d_host.t2d_num= host;
t2d_port.t2d_num= port;
t2d_path.t2d_num= path;
t2d.t2d_num= 1;
return 0;
```

6.5.4 on_watchlist

```
/* ------------------------------------------------------------------ */
on_watchlist: parse arg host, port
do i=1 to sc_num;
  if sc_host.i=host & sc_port.i=port then return 1;
end;
return 0;
```

6.5.5 is2b_done

```
/* ------------------------------------------------------------------ */
is2b_done: parse arg host, port, path
do i=1 to t2d_num;
  if t2d_host.i=host &,
    t2d_port.i=port &,
    t2d_path.i=path then return 0;      /* directory is put on         */
                                        /* work list just once         */
end;
return 1;
```

6.5.6 get_item

```
/* ------------------------------------------------------------------ */
get_item: parse arg host, port, path

if relax/='' then 'SLEEP' relax;
relax= def_relax;

'GOPHFTCH' tmpfile '0' host port path

dirs_scanned= dirs_scanned+1;
```

```
/* the file is already (or should be) finis'ed by GOPHFTCH */
'FINIS' tmpfile
address '' 'STATE' tmpfile
if rc/=0 then return 2; /* not ok, but dont try it again */
dirs_ok= dirs_ok+1;

MENU.0=0;
'EXECIO * DISKR' tmpfile '(FINIS STEM MENU.'
'EXECIO' MENU.0 'DISKW' outfile '(FINIS STEM MENU.'
lines_scanned= lines_scanned + menu.0;

do mi=1 to MENU.0;
  parse var MENU.mi name '05'x path '05'x host '05'x port '05'x .
  type= substr(name,1,1);
  name= substr(name,2);
  /* say 'type='type 'name='name 'host='host 'port='port */
  if type=1 then do;
    watchit= on_watchlist(host, port);
    if watchit=1 then do;
      is2bd= is2b_done(host, port, path);
      if is2bd=1 then x= add_item(host, port, path);
    end;
  end;
end;
return 0;       /* 0 ... success             */
                /* 1 ... needs to be done    */
                /* 2 ... error               */
```

7 Examples

7.1 Module PROFSRV EXEC

```
#erase PROFSRV EXEC
#module PROFSRV EXEC
/* ------------------------------------------------------------------ */
/* 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 .... */
```

7.2 Module GOPHERD SINDEX

```
#erase GOPHERD SINDEX
#module GOPHERD SINDEX
0About this Gopher Installation 0/about #<CL_LOCALHOST># 70
1The mother of all Gophers (U/Minnesota) gopher.tc.umn.edu 70
1University of Economics, Vienna gopher.wu-wien.ac.at 70
```

7.3 Module GOPHERD SSELECT

```
#erase GOPHERD SSELECT
#module GOPHERD SSELECT
GOPHERD.ABOUT.B gg * 0/about
```

7.4 Module GOPHERD SEAI

```
#erase GOPHERD SEAI
#module GOPHERD SEAI
* This is an example GOPHERD EAI file for this installation
```

7.5 Module GOPHERD SGROUPS

```
#erase GOPHERD SGROUPS
#module GOPHERD SGROUPS
* This is an example GOPHERD GROUPS file, an almost empty one ...
schnurli 137.208.16.39
wu-wien  137.208.
```

7.6 Module GOPHERD SDSKGRP

```
#erase GOPHERD SDSKGRP
#module GOPHERD SDSKGRP
* This is an example GOPHERD DSKGRP file
A1      1       schnurli
B1      1       *
```

7.7 Module GOPHERD SABOUT

```
#erase GOPHERD SABOUT
#module GOPHERD SABOUT
This is an example GOPHERD ABOUT file for this installation
```

8 Postinstaller

8.1 Module GOPHINST EXEC

```
#erase GOPHINST EXEC
#module GOPHINST EXEC
/*********************************************************************/
/* File GOPHINST EXEC #<RELEASE># #<RELEASE_DATE># #<RELEASE_TIME># */
/*                                                                   */
/* The postinstaller module for the VieGOPHER system                 */
/*                                                                   */
/* written:       1993-05-02: <Gerhard.Gonter@wu-wien.ac.at>         */
/* latest update: 1993-05-26                                         */
/*********************************************************************/
parse arg modus par

upper modus;

f_gopherc=  0;
f_rxsocket= 0;
f_rextcpip= 0;

sendfix= 'NO';
compact= 'NONE';
compact_tmp1= 'REXXCPCX.CPT.A';
compact_tmp2= 'REXXCPCX CPT A';
srv_wakeup_system= 'NO';
erase_unneeded_files= 0;
copy_sample_files= 0;

/* checkout calling options */
select;
  when modus='AUTO1' then do;
    parse var par srv_tcp cl_tcp srv_wakeup_system sendfix compact .
    f_gopherc=  1;
    select;
      when SRV_TCP='RXSOCKET' then f_rxsocket= 1;
      when SRV_TCP='REXTCPIP' then f_rextcpip= 1;
      otherwise nop;
    end;
    select;
      when CL_TCP='RXSOCKET' then f_rxsocket= 1;
      when CL_TCP='REXTCPIP' then f_rextcpip= 1;
      otherwise nop;
    end;
    address '' 'STATE REXXCPCX MODULE'
    if rc/=0 then compact= 'NONE';
    upper compact;
    copy_sample_files= 1;
    erase_unneeded_files= 1;
  end;
  when modus='MAKE' then do;
    upper par;
    do while par/='';
      parse var par p1 par
      upper p1;
      if p1='GOPHER' then f_gopherc= 1;
      if p1='SAMPLES' then copy_sample_files= 1;
```

```
    end;
  end;
  otherwise say 'unknown postinstallation mode' modus par
end/*select*/;

/* check or prepare a gopher exec */
if f_gopherc=1 then do;
  say '----- checking GOPHER EXEC -----'
  gc= check_gopher('GOPHER EXEC', 'GOPHERC');
  select;
    when gc=0 then do;
      say 'Great! a customized GOPHER EXEC file exists.'
    end;
    when gc=1 then do;
      say 'Apparently, no customized GOPHER EXEC file exists sofar.'
      say "I'm going to create one, please customize as needed later."
      x= create_gopher('GOPHER EXEC A');
    end;
    when gc=2 then do;
      say 'Apparently, you have an early version of GOPHER.EXEC.'
      say 'In this version of VieGOPHER, the file is called GOPHERC.EXEC'
      say 'and should be called by a customized program called GOPHER.'
      say ''
      say 'Please, create GOPHER.EXEC by hand or run:'
      say '  GOPHINST MAKE GOPHER';
      say "but don't forget to remove the old version before."
    end;
    otherwise do;
      say 'checking for gopher exec returned:' gc;
    end;
  end/*select*/;
end;

if f_rxsocket=1 then do;
  say '----- checking RXSOCKET MODULE -----'
  address '' 'LISTFILE RXSOCKET MODULE * (ALLOC DATE';
  if rc/=0 then do;
    say 'You need the file RXSOCKET MODULE in order to run this'
    say 'gopher installation properly.'
    say 'The file can be obtained from LISTSERV@CUNYVM.bitnet.'
    say ''
    say 'Shall I order it right now? (YES or NO)'
    pull answ .
    if answ='YES' then 'TELL LISTSERV at CUNYVM get RXSOCKET MODULE'
  end;
end;

if f_rextcpip=1 then do;
  say '----- checking REXTCPIP MODULE -----'
  address '' 'LISTFILE REXTCPIP MODULE * (ALLOC DATE';
  if rc/=0 then do;
    say 'You need the file REXTCPIP MODULE in order to run this'
    say 'gopher installation properly.'
    say 'The file can be obtained from LISTSERV@DEARN.bitnet.'
    say ''
    say 'Shall I order it right now? (YES or NO)'
    pull answ .
    if answ='YES' then 'TELL LISTSERV at DEARN get REXTCPIP MODULE'
  end;
```

```
end;

if srv_wakeup_system/='NO' then do;
  select;
    when srv_wakeup_system='YWAKEUP' then do;
      say '----- checking YWAKEUP MODULE -----'
      address '' 'LISTFILE YWAKEUP MODULE * (ALLOC DATE';
      if rc/=0 then do;
        say 'You need the file YWAKEUP MODULE in order to run this'
        say 'gopher installation properly.'
        say 'The file can be obtained from LISTSERV@DEARN.bitnet.'
        say ''
        say 'Shall I order it right now? (YES or NO)'
        pull answ .
        if answ='YES' then 'TELL LISTSERV at DEARN get YWAKEUP MODULE'
      end;
    end;
    when srv_wakeup_system='WAKEUP' then do;
      say '----- checking WAKEUP MODULE -----'
      address '' 'LISTFILE WAKEUP MODULE * (ALLOC DATE';
      if rc/=0 then do;
        say 'You need the file WAKEUP MODULE in order to run this'
        say 'gopher installation properly.'
        say 'The file should be delivered by IBM as part of',
            'the operating system.'
      end;
    end;
    otherwise do;
      say 'warning: unknown wakeup system:' srv_wakeup_system
    end;
  end/*select*/
end;

if sendfix='YES' then do;
  say '----- sending VIEGOPHR FIX to GONTER@AWIWUW11.bitnet -----'
  'SENDFILE VIEGOPHR FIX GONTER AT AWIWUW11';
end;

if compact/='NONE' then do;
  if compact='ALL' | index(compact,'C')>0 then do;
    say '----- compacting client modules -----'
    x= do_compacting('EXEC', 'GOPHERC  GOPHFTCH');
    x= do_compacting('XEDIT','GOPHER   GOPHRING GOPHRNG2 GOPHRNG3');
    x= do_compacting('XEDIT','GOPHSCRN GOPHSRCH GOPHXBMC GOPHXBMI');
    x= do_compacting('XEDIT','GOPHXBMS GOPHXFTC GOPHXQIT GOPHXSVI');
  end;
  if compact='ALL' | index(compact,'S')>0 then do;
    say '----- compacting server modules -----'
    x= do_compacting('EXEC', 'GOPHERDD');
  end;
/* @@@   ------------------------------------------------------ @@@ */
/* @@@   if compact='ALL' | index(compact,'U')>0 then do;       @@@ */
/* @@@     these modules are not used any longer                @@@ */
/* @@@     say '----- compacting utility modules -----'         @@@ */
/* @@@     x= do_compacting('EXEC', 'GLOBALXV GLOBALVV');       @@@ */
/* @@@   end;   @@@ */
/* @@@   ------------------------------------------------------ @@@ */
  if compact='ALL' | index(compact,'T')>0 then do;
    say '----- compacting tools modules -----'
```

```
    x= do_compacting('EXEC', 'COOKIE   EDINDEX FINGER FINGERXX');
    x= do_compacting('XEDIT','EDINDEX');
  end;
end;

if erase_unneeded_files=1 then do;
  say '----- erasing unneeded files -----';
  x= xerase('GLOBALVV EXEC A');
  x= xerase('GLOBALXV EXEC A');
  x= xerase('GOPHRNG2 XEDIT A');
  x= xerase('FDNS EXEC A');
end;

if copy_sample_files=1 then do;
  say '----- copying sample files -----';
  x= copy_sample('GOPHERD','SINDEX','INDEX',' (root index)');
  x= copy_sample('GOPHERD','SSELECT','SELECTOR',' (selector strings)');
  x= copy_sample('GOPHERD','SGROUPS', 'GROUPS',' (group defs)');
  x= copy_sample('GOPHERD','SDSKGRP', 'DSKGRP',' (disk group defs)');
  x= copy_sample('GOPHERD','SEAI','EAI',' (embedded apps desc)');
  x= copy_sample('GOPHERD','SABOUT','ABOUT','');
end;

exit(0);

/* ------------------------------------------------------------------ */
check_gopher: parse arg mod, seq;

/* all files of the same name on all disks, if not specified */
parse var mod x1 x2 x3 .
if x3/='' then x3= '*';
mod= x1 x2 x3;

q= queued();
address '' 'LISTFILE' mod '(ALLOC DATE';
if rc/=0 then return 1;
q= queued()-q;

do j=1 to q;
  pull x1 x2 x3 .
  modn.j= x1 x2 x3
end;

do j= 1 to q;
  'EXECIO * DISKR' modn.j '(FINIS STEM PGM.'
  do i=1 to PGM.0;
    if index(PGM.i,seq) > 0 then return 0;
  end/*i*/;
end/*j*/;
return 2;

/* ------------------------------------------------------------------ */
create_gopher: parse arg mod;
"EXECIO 1 DISKW" mod "(STRING /* customize this file as needed. */"
"EXECIO 1 DISKW" mod "(STRING parse arg pars"
"EXECIO 1 DISKW" mod "(STRING 'GOPHERC' pars"
'FINIS' mod;
return 0;
```

```
/* ------------------------------------------------------------------ */
do_compacting: parse arg type, list;
say 'compacting ('type')' list
do while list/='';
  parse var list fi list
  address '' 'STATE' fi type A
  if rc=0 then do;
    'REXXCPCX' fi'.'type'.A' compact_tmp1
    'ERASE' fi type 'A'
    'RENAME' compact_tmp2 fi type 'A';
  end;
end;
return 0;

/* ------------------------------------------------------------------ */
copy_sample: parse arg fnm, ft1, ft2, desc
address '' 'STATE' fnm ft2 'A'
if rc/=0 then do;
  say "copying sample file"desc" '"fnm ft1"' to '"fnm ft2"'"
  'COPYFILE' fnm ft1 'A' fnm ft2 'A'
end;
return 0;

/* ------------------------------------------------------------------ */
xerase: parse arg fnm
address '' 'STATE' fnm
if rc=0 then 'ERASE' fnm
return 0;


# Finally, this is the call to the postinstaller program, delivered
# within the gopher product file.

#call GOPHINST AUTO1 SRV_TCP CL_TCP SRV_WAKEUP_SYSTEM SENDFIX Compactor
```