FUNETNJE 3.x

NJE-emulator for UNIX systems

DRAFTDRAFT**DRAFT**DRAFT**DRAFT**DRAFT**DRAFT**DRAFT**

Matti Aarnio
Finnish University and Research Network, FUNET

1 Support of the FUNETNJE package

Academic version of this software is available from:

```
ftp://ftp.funet.fi/pub/unix/networking/bitnet/funetnje-datecode.tar.gz
```

Academic version of this software comes AS-IS without any guarantees of working for any particular purpose, or that it is bug-free, or that it won't cause trouble, loss of valuable data, etc. – for short, you are essentially on your own.

Like all Academic software, there might come up a bugfix, or improvement sometimes, but such effort being non-funded, and without any schedules, there are no guarantees of anything.

User-community is welcomed to subscribe mailinglist:

```
$ Mail mailserver@nic.funet.fi
Subject: xxxx
subscribe funetnje Your Name
```

Then corresponding via address: *funetnje@nic.funet.fi*

Remember: LIST administrativia goes via mailserver!

When registering this software to your *BITNET*-node, I request that you use name: **FUNETNJE**

If you want to pay for hand-holding service/timely bugfixing, special customizations, etc. please contact author:

- <*mea@nic.funet.fi*>

2 History of the FUNETNJE package

"Credit where the credit is due to"

Original version of this software is known as *HUJI-NJE*, which is VAX-VMS software for implementing BITNET-II, and several BISYNC modes of the NJE network. It was made at the *Hebrew University of Jerusalem* (HUJI), by Mr. *Yehavi Bourvine* over the years 1988-1993.

Sometime around 1990-1991 FUNET archive server was decided to be connected to the EARN, and the best (= free in source form) technology for it turned to be HUJI-NJE.

The original HUJI-NJE is monsterous monolithic program which has built-in processing of incoming email, and translation of files (in and out) in between ASCII and EBCDIC.

At first Matti Aarnio <mea@nic.funet.fi> did minor porting of the program to get it to work on a Sun4/330 server (such as *nic.funet.fi* was back then), which did surface several problems in between VAX, and other architectures.

Latter the HUJI-people lost their interest on the system, and when no radical improvements were on sight, begun the more serious porting of the system to Sun4 machine.

Then on Summer-93 came a contact from Mr. Gerald Hanusch <K000165@ALIJKU65>, who wanted to improve the then-present version of FUNETNJE to be more complete in respect of its emulation of the RSCS for replacing their aging VM/XA CMS machine – they had administrative systems on another MVS machine, and communication to it was to be preserved, although the VM/XA CMS machine was turned off.

Over the months the system improved, and it was considered that all the necessary major parts of the NJE-emulator were functional on January-94.

Up until then the software developement was on "fun to do" basis, and beside the actual job of the parties involved.

See previous chapter about *Support* for more about the supported version.

3 Introduction to NJE-networking

Traditionally NJE-networking happens (did happen) in between two IBM mainframes via BISYNC cable on 4800/9600 bps speed.

Used protocols do vary, but most widely used ones have multiple interleaved streams etc. in them, and they keep a tab on line condition by timeouting, if acknowledgements don't appear quick enough. The protocols also have a sort of CRCs in them for every "packet" they send over the wire.

Things that are possible with NJE-network are:

Unsolicited file transfer
: Sending files to the recipients without them being aware of files coming to them, nor in fact asking any authorization prior sending it to them.

  Especially this contradicts the way how the FTP works.

  This happens in *Store-and-Forward*-manner, that is, when the file is sent over one link, responsibility for it is transferred to the next node, and link-sender can delete it from its own spool.

Remote Job Entry
: Sending batch-jobs to remote machine for execution, especially to MVS systems.

Nodal Messaging
: Sending short (max 120 chars) lines of messages to recipients at other nodes.

With these basic facilities it is possible to build extended services, like:

- Remote printing
- Email
- File transfer
- Remote Job Execution (Batch processing)
- Interactive discussion one line at the time via NMR-messages
- Interactive control of servers accepting NMR-messages

This networking mechanism allows to have simple phone-line quality links to be used to transport huge amounts of data over long distances. It may be somewhat slow at the times, but it works. . .

An entry-level BITNET connectivity requires capability to communicate via synchronous BISYNC interface, although these days BITNET-II technology has largely surplanted it. See on next chapter.

There exists even ASYNC NJE, although that is more like experimental kludge for hackers, than serious production tool. (Not supported by the FUNETNJE.)

Names in the BITNET are at most 8 characters long. This applies both to the users, and the machines (*nodes*).

Routing in the BITNET requires every machine to know every other, and therefore there exists fairly large bureaucracy for keeping the linkage information up to date.

Full routing updates are done in semi-monthly manner, which updates can usually be incorporated into the system with minimal fuzz, or in case of FUNETNJE, automatically.

There are email-gateways in between the BITNET, and the rest of the world. Email is transported inside the BITNET with five methods:

defrt1 Target system has no mailer, all email must be sent to the users directly. Normally a mail-file is sent in PUNCH-80 format.

bsmtp3 Target system has a mailer, which can receive Batch-SMTP-file containing the email. Format of the file is PUNCH-80.

bsmtp3rfc Like **bsmtp3**, but addresses in RFC-822 format instead of BITNET node style.

bsmtp3nd Target system has a mailer, which can receive Batch-SMTP-file containing the email. Format of the file can be PUNCH-80, or NETDATA. In case the email contains longer lines, than 80 characters, this is recommended method, if receiver system can accept it.

bsmtp3ndrfc
Like **bsmtp3nd**, but addresses in RFC-822 format instead of BITNET node style.

For more information about these tags, see below about mailer configuration.

To familiarize him-/herself with NJE-networking, the reader should also have access to the IBM manual '*Network Job Entry Formats and Protocols for System/370 Program Products*', **GG22-9373-02** or **SC23-0070-01** (or later).

4 Structure of Second Generation BITNET on virtual links

BITNET was a traditional wires+modems network, until the NSF-net began to grow in late 1980's.

When it became evident, that having separate wires for BITNET, and for other networking activities (TCP/IP, NSF-net) is asking to keep old separate lines in existance with new high-speed Internet, multiple solutions were sought after for replacing physical wires with virtual ones.

In 1986-1989 Ira Fuchs, Peter Olenick et.al. (Princeton University) developed so called BITNET-II protocol, with which it became possible to replace dedicated physical wires with virtual TCP/IP connections over the Internet.

The protocol, and its reference implementation are described on `BRFC0002 TEXT` available at least from ftp.funet.fi:

```
ftp://ftp.funet.fi/pub/doc/netinfo/CREN/brfc0002.text
```

via anonymous FTP.

Soon after, (in 1990), Lee Varian, Peter Olenick, and Michael Gettes (all from Princeton University) made a BITNET restructuring proposal in which they divided fine-grained threaded mesh of the former BITNET into 9 regions.

From their `bit2plan.proposal`:

February 1, 1990
Revised September 21, 1990

Proposal to Restructure the Network Supporting BITNET

Historically the physical and logical networks supporting BITNET have been structured on leased communications lines connecting member sites. New sites connect to the network based upon an existing BITNET member's willingness and ability to support the request for connection. The cost of the leased communications line, which is paid for by the new BITNET site, is also a consideration in choosing which existing BITNET sites are possible connection points. The cost of the communications line may be different based on factors of distance and tariff considerations. This method of connecting new sites has worked quite well, but the lack of a formal structure for the network is creating limitations which impair the efficiency and growth of BITNET. The lack of a structure increases the complexity of routing table generation and hampers efforts to implement network management tools. The ability to create BITNET links by using the national and regional IP networks increases the confusion over where BITNET connections should be made.

The idea of reorganizing BITNET into a more formal structure has been discussed for a number of years, but the costs of the additional leased lines needed to implement such a plan made the project economically infeasible. Now, BITNET has the ability to use the national and regional IP networks; a plan to restructure BITNET by using the IP networks appears to be cost effective and implementable. This plan could also be implemented using private communications facilities and IP routers instead of using the national and regional networks. The costs of an implementation using a private IP network would need to be carefully considered.

Restructuring Proposal

The restructuring is based on the concept of 'regionalization', the separation of the network into geographic areas or regions. Each region would have two 'core' sites. Each core site would have RSCS-over-IP connections to every other core site. The

core sites would form a 'backbone'. The national and regional IP networks are the physical facilities that would be used by the core sites to form the BITNET backbone. By generating appropriate BITNET routing tables, the number of nodes and the amount of traffic handled by the core sites for a given region can be statically balanced. Within a region, the core sites could connect to a number of 'mid-level' sites, again by use of RSCS-over-IP. This type of structure has the ability to provide an alternate path into a region if a core site were out of service. The member sites or 'end' sites within a region could connect to the mid-level sites. Traditional leased line connections may exist at any level within the structure but these types of connections will continue to have limitations. That is, if a host with traditional leased lines is down, no other path may exist to the sites supported by that leased line.

. . .

Benefits of the restructuring

The purpose of the regionalization is to impose a structure on the logical network supporting BITNET. This structure will reduce the burden on the current hub sites by decreasing the number of files which must transit these sites. Overall network service will be improved because the number of 'hops' a file must take to reach its destination will be reduced or be no greater than in the current BITNET topology. The impact on BITNET when a key BITNET node or Internet connectivity fails will be reduced because of the increased number of connections between core sites. As the intra-regional mid-level structure develops, the ability of the core sites to dynamically reroute traffic around a disabled core node will provide improved network access. The three level (core, mid-level, end site) structure of the region can be expanded to include additional levels and paths as needed within the region, thereby providing for dynamic rerouting within a region as well.

5 Installation of FUNETNJE package

Installation of this package is fairly straight-forward by editing proper parts from to be activated from the `Makefile`, and editing sample configuration files for your local machine.

This system has two hard-wired files, which can be overridden by editing `site_consts.h`-file, or defined in `Makefile`:

/etc/funetnje.cf
: Main configuration file telling where rest of the system resides

/etc/funetnje.pid
: The "pid file" giving easy pointer to the running emulator program

Via `/etc/funetnje.cf`, and its extensions (`msg-exit.cf`, `file-exit.cf`) it is possible to define outgoing, and incoming spool directories, et.al. Only notable detail is that all those directories **must** reside on same disk partition, as files are moved from within emulator spool to user spool with *link(2)*-, and *unlink(2)*-system calls. (Or *rename(2)*, if it exists.)

For inter-module access control this software uses one UNIX group, which is used to *set-gid* some programs, and which is needed for system administrators so that they can run the control program without a need to become super-users, and also without a need to set-gid the control-panel (`ucp`).

Things to decide:

- Name and number of the 'bitnet' programs group (into your `/etc/group`). The `Makefile` has variable *NJEGRP* for it.
- Directories for your BITNET transmission spool, and user spool. *Important: those must be on same partition!*

  Example configurations have them as:

  - *QUEUEDIR* `/var/spool/bitnet`
  - *USERSPOOL* `/var/spool/bitspool`
- Location of various programs, `Makefile` has variables: *MANDIR*, *LIBDIR*, *BINDIR*, and *ETCDIR*.

Choosing proper defines for the compile of the software, especially your platform dependent pre-processor constants:

-D_POSIX_SOURCE
: Use on true `POSIX.1` systems

-DHAS_PUTENV
: If your system has SysV-like `putenv(3)` to set environment variable. When this is not defined, a BSD-like `setenv(3)` is assumed

-DHAS_LSTAT
: Use when your system has `lstat(2)` in addition to the usual `stat(2)`

-DUSE_SOCKOPT
: Use when your system has a working `setsockopt(2)` for `SO_RCVBUF`, and `SO_SNDBUF`. Most systems do! This is a performace option, which **might** help on busy systems

-DNBCONNECT
: The system can do a `connect(2)` on a socket, which has been flagged as non-blocking. Without this connection to a system which is unreachable would cause system to hang for synchronous timeout... (Superceded with **NBSTREAM**)

-DNO_NBSTREAM
If for some reason you want to disable the non-blocking stream code that uses following two defined, define this.

-DNBSTREAM
This compilation option allows for more complete non-blocking operations, including actual `write(2)`s. Defined in `consts.h`, unless **-DNO_NBSTREAM** is defined.

-DUSE_XMIT_QUEUE
Another flag which is necessary for successfull operation of NBSTREAM-code. (Without it the NBSTREAM-code causes bug_check()s..) Defined in `consts.h`, unless **-DNO_NBSTREAM** is defined.

-DNO_GETTIMEOFDAY
System internal time retrieval for various delay/speed-analysis uses uses (per default) BSD-derived(?) `gettimeofday(2)`-call, but if your system does not have it, define this, and `time(2)` will be used.

-DMAILERNAME="..."
To override the default *MAILERNAME* (= "MAILER") in the `bmail.c` source.
For normal use the `bmail`'s "-u"-option does the same

-DCONFIG_FILE=...
Override `consts.h` definition of system config file name

-DPID_FILE=...
Override `consts.h` definition of emulator process-id file name

-DBSD_SIGCHLDS
Uses `BSD`-style `SIGCLD/SIGCHLD` handling where an signal trapper program is required to process the exited child information. On `SysV`-systems it is safe (it seems) to just ignore `SIGCLD` and that way to get rid of the zombied child.

Some compiler switches:
-fno-builtin
A GNU-CC option telling it, that builtin accelerated operations should not be used. (Alignment may cause problems on some cases!) However we have a fairly high confidence that we have eliminated every case, where inlining a `memcpy(3)` could cause problems. Try without it! (Possible problems manifest themselves as coredrops due to SIGBUS, as some memory accesses may become unaligned..)

Some more esoteric tricks:
-DDEBUG
If you really want to see what goes on... The amount of DEBUG-code is decreasing all the time, and frankly we haven't compiled the code with '-DDEBUG' for last 100 or so versions, we have just added 'logger()'-calls into it at places needing it, and adjusted *LOGLEVEL* to get them into *LOGFILE*, and then commented or removed those loggers out again.. (Or used debuggers to get into the business..)

-DUSE_ZMAILER -I/usr/local/include
For `mailify.c` – for tight integration with ZMailer MTA. (Expects ZMailer's `libzmailer.a`, and `zmailer.h` to be available. See ZMailer's `zmailer(3)` for further information)

-DUSE_OWN_PROTOS
If the "prototypes.h" -function prototypes for various things can for your system, and your system does not have ANSI-headers of its own, then use this

Several pre-tested compiler setups are also available in the `Makefile`.

6 Configuration of FUNETNJE package

6.1 Runtime main configuration-file: `funetnje.cf`

Here is a sample `/etc/funetnje.cf` -file:

```
*
*       Configuration file for FUNETNJE program
*

NAME            FINFILES                      Node primary name
*ALIAS          HAMSTER                       Optional alias for this node
*IPADDRESS      nic.funet.fi                  Node primary IP address
IPADDRESS       128.214.6.100                 used on link signon msgs
QUEUE           /usr/spool/bitnet             Emulator spool directory
CMDMAILBOX      FIFO /usr/spool/bitnet/.cmdpipe  IPC Fifo -path
#CMDMAILBOX     SOCK /usr/spool/bitnet/.cmdpipe  IPC Socket -path
#CMDMAILBOX     UDP 128.214.6.100 175            IPC IP-address + port number
LOG             /usr/adm/bitnet.log           Emulator log-file
LLEVEL          1                             Logging level.  Smaller==less
RSCSACCT        /usr/adm/bitnet.acct          Define for IBM RSCS-like account file
* this table is a binary file.
TABLE           /usr/local/lib/nje/finfiles.routes   NJE route database
* EBCDIC<->ASCII tables, (optional), see ebcdictbl(5)
*EBCDICTBL      /usr/local/lib/nje/ebcdictbl
INFORM          MEA@FINFILES                  1+ users monitoring line
INFORM          ROOT@FINFILES                     states
FILEEXITS       /usr/local/lib/nje/file-exit.cf      Defines  FILE-EXITs
MSGEXITS        /usr/local/lib/nje/msg-exit.cf       Defines  MSG-EXITs
DEFFORM         STANDARD                      NJE 'FORM' default
*DEFAULT-ROUTE  FINHUTC                       Optional default route.  With this
                                              the TABLE-file can be left empty
                                              but it must exist!

*  Example link to a VAX/VMS system with UCX, and JNET
LINE 0 FIGBOX                           Key-line starting link definition
 TYPE           UNIX_TCP                Only supported value is  UNIX_TCP
 BUFSIZE        4096                    VMNET TTB size
 IPPORT         500                     Opposite end's VMNET's IP port
* TCPNAME       figbox.funet.fi         Opposite end's name - as a comment
 TCPNAME        128.214.6.7              .. or IP number ..
 MAX-STREAMS    7                       Number of parellel streams
 TCP-SIZE       8192                    How much to block before write
 TIMEOUT        3                       .. and how long to wait to flush par-
tial block?

*  Example link to a Linux-PC running FUNETNJE
LINE 1 ALIJKU65
 TYPE           UNIX_TCP
 BUFSIZE        1024
 IPPORT         175             The official VMNET port
*TCPNAME        alijku65.edvz.uni-linz.ac.at
```

```
 TCPNAME        140.78.4.34
 MAX-STREAMS    7
 TCP-SIZE       3100            Lower than usual - network problems
 TIMEOUT        3

*  Example link to an IBM VM/SP RSCSv3 system with VMNET
LINE 2 FINHUTC
 TYPE           UNIX_TCP
 BUFSIZE        1024
 IPPORT         175
*TCPNAME        finhutc.hut.fi
 TCPNAME        130.233.224.4
 MAX-STREAMS    7
 TCP-SIZE       8192
 TIMEOUT        3

*  Example link to a Convex C3480 running FUNETNJE
LINE 3  FINFAN
 TYPE           UNIX_TCP
 BUFSIZE        8192
 IPPORT         175
 TCPNAME        convex.csc.fi
 MAX-STREAMS    7
 TCP-SIZE       8192
 TIMEOUT        3
```

6.2 Incoming files processing configuration file: `file-exit.cf`

This is *wide* format configuration file (like they all), and can be used to define what is done with various incoming files.

Comment lines are those that start with "#" or ";" on column 1. The "*" is a wild-card needed on patterns.

Over-wide lines are wrapped *in this document* with \-characters at the end of previous line, and at the begin of next line. Such wrapping is not allowed in the real configuration file!

```
#  FILE-EXIT.CF -- Configure file handling on FUNETNJE
#
# Rule for spool disposition path, defines also system default
# user spool directory for those who don't have specially set
# something else.
# Possibilities:    ~/bitspool  -- real user's only! (~/ == users home)
#                   /usr/path/path/ -- append 'touser' as subdir,
#                                       it is spool dir.
#                                      Valid for Exits and real users.
#                   /usr/path/path  -- explicite directory.
#                   /dev/null       -- explicite file (special case).
#       When directory isn't present, it is built from upmost present
#       level down to users bottommost level hmm...
#   Question about ownership of directory/files...
#     Real users:  real protections, programs start with setuid() user.
#         Exit users:  POSTMAST  (exits start as root anyway.)
#         Exited reals: real protections, programs start with setuid()
#                       user.
```

```
Spool-Dir:      /usr/spool/bitspool/
Postmast-Dir:   /usr/spool/bitspool/POSTMAST

# Now list of things to match and then what to do
# To do keywords:       DISCARD to /dev/null.
#                       KEEP    just so.  Into default or given spool.
#                       NOTIFY  send an NJE message to someone.
#                       RUN     starts arbitary program with arbitary
#                               arguments telling about file location
#                               and its properties.
#                               If fails, well..

# Defining  SpoolDir  which shall not be attached ToUser must not be done
# with trailing "/", (and vice versa)
#       /usr/spool/bitnet/SYSIN-JOB

# Exit table begin keyword:
Exit-Table:

# Args:
# touser8 tonode8 fname8   ftype8  pun? class fruser8  frnode8  dist8    \
\ SpoolDir                       action  ExtraArguments
#
# Several special handlings (tests/illustriations)
#
#MEA     FINFILES *        *        *   *     HKS      SEARN    *        \
\ default                        DISCARD
NOBODY   FINFILES *        *        *   *     *        *        *        \
\ default                        DISCARD
FOOBAT   FINFILES *        *        *   *     *        *        *        \
\ default                        RUN   /usr/local/lib/nje/transfer \
\ MEA@$TONODE $SPOOL
ECHO     FINFILES *        *        *   *     *        *        *        \
\ default                        RUN   /usr/local/lib/nje/transfer \
\ NOBODY@$FRNODE $SPOOL
#LPR     FINFILES *        *        *   *     *        *        *        \
\ default                        RUN   /usr/local/lib/nje/rprint   \
\ $SPOOL lpr $FRUSER $FRNODE $TOUSER $FID
#
#  Define what we do with the SYSINs
#
*        *        *        *        SYSIN *   *        *        *        \
\ /usr/spool/bitnet/SYSIN-JOB/    RUN   /usr/local/lib/nje/sysin $SPOOL
#
#  Define a MAILER for our node!
#  And also handle directly sent mail from nodes without mailers...
#
MAILER   FINFILES *        *        PUN M   *        *        *         \
\ default                        RUN   /usr/local/lib/nje/mailify $SPOOL
*        *        *        MAIL     PUN M   *        *        *         \
\ default                        RUN   /usr/local/lib/nje/mailify $SPOOL
```

```
#
# NETINIT is pseudo id for automatically re-generating routing tables
# This is done with  netinit.sh  which must be configured by hand.
#
NETINIT  FINFILES *        *        *   *   *        *        *          \
\ default                          KEEP
#
# Finally  a 'catch them all' default handling case.
#
*        *        *        *        *   *   *        *        *          \
\ default                          KEEP
```

6.3 Incoming NMR's processing configuration file: `msg-exit.cf`

This is *wide* format configuration file (like they all), and can be used to define what is done with various incoming NMR's.

Comment lines are those that start with "#" or ";" on column 1. The "*" is a wild-card needed on patterns.

Over-wide lines are wrapped *in this document* with \-characters at the end of previous line, and at the begin of next line. Such wrapping is not allowed in the real configuration file!

```
#
# MSG-EXIT.CF  for FUNETNJE
#
# Some first ideas about patterns, et.al.:
#
# Actions:  CMD: BUILTIN, RUN
#           MSG: BRCAST, DISCARD, RUN, (PIPE)
# Argument tokens for 'RUN': TOUSER, TONODE, FRUSER, FRNODE, TEXT, ARGS
# Arguments for 'BUILTIN': "HELP", "HARDCODED", "ERROR"/ERROR msg-string",
#                          "ALIAS remap-pattern"
# Patterns work as follows:
#      TO*KEN -- match (only) any of input words: TO TOK TOKE TOKEN
#      *      -- match any token, (but not blank)
#      **     -- match any number of tokens, including nothing.
# Remap-patterns work as follows:
#      $nn    -- substitute nn:th text token into this place
#      ANYSTRING -- copy it verbatim
#

CmdHelpFile: /usr/local/lib/nje/cmd-help.txt

#$TOUSER $TONODE  $FRUSER  $FRNODE  C pattern    ACTION args
# Actually TOUSER is not tested on commands, it will be blank anyway..
.        FINFILES *        *        C "HIL*FE"   \
\ BUILTIN HELP English quick document is available via 'HELP' command.
.        FINFILES *        *        C "SOS"      \
\ BUILTIN HELP English quick document is available via 'HELP' command.
.        FINFILES *        *        C "H*ELP"    BUILTIN HELP
.        FINFILES *        *        C "M * * * **"       BUILTIN HARDCODED
.        FINFILES *        *        C "MSG * * * **"     BUILTIN HARDCODED
.        FINFILES *        *        C "M*SG **"          \
\ BUILTIN ERROR Too few arguments for the MSG
```

```
.        FINFILES *        *        C "CMD * * **"       BUILTIN HARDCODED
.        FINFILES *        *        C "CMD **"           \
\ BUILTIN ERROR Too few arguments for the CMD
.        FINFILES *        *        C "Q*UERY SYS*TEM S" BUILTIN HARDCODED
.        FINFILES *        *        C "Q*UERY SYS*TEM"   BUILTIN HARDCODED
.        FINFILES *        *        C "Q*UERY STAT*S"    BUILTIN HARDCODED
.        FINFILES *        *        C "Q*UERY *"         BUILTIN HARDCODED
.        FINFILES *        *        C "Q*UERY * Q*UEUE"  BUILTIN HARDCODED
.        FINFILES *        *        C "CPQ*UERY N*AMES"  BUILTIN HARDCODED
.        FINFILES *        *        C "CPQ*UERY U*SER *" BUILTIN HARDCODED
.        FINFILES *        *        C "CPQ*UERY U*SER"   BUILTIN HARDCODED
.        FINFILES *        *        C "CPQ*UERY LOG"     BUILTIN HARDCODED
.        FINFILES *        *        C "CPQ*UERY CPU"     BUILTIN HARDCODED
.        FINFILES *        *        C "CPQ*UERY CP*LEVEL" BUILTIN HARDCODED
.        FINFILES *        *        C "CPQ*UERY IND*ICATE" BUILTIN HARDCODED
.        FINFILES *        *        C "CPQ*UERY T*IME"   BUILTIN HARDCODED
.        FINFILES *        *        C "CP*QUERY T*IME"   BUILTIN ALIAS CPQ $2
.        FINFILES *        *        C "EC*HO **"     \
\ RUN /usr/local/bin/send -s $FRUSER@$FRNODE *CMD was '$TEXT'
# The default
.        FINFILES *        *        C "**"       BUILTIN ERROR

#$TOUSER $TONODE  $FRUSER  $FRNODE  M ACTION args
#                "." == FRUSER is blank, that is from some 'SYSTEM'..
VMNET    *        *        *        M  DISCARD
MAILER   *        *        *        M  DISCARD
MAILSERV FINFILES .        *        M  DISCARD
MAILSERV FINFILES *        *        M  \
\ RUN  /usr/local/lib/mailserver/nje-msg $FRUSER $FRNODE $TEXT
LISTSERV FINFILES .        *        M  DISCARD
LISTSERV FINFILES *        *        M  \
\ RUN  /usr/local/bin/send -u MAILSERV $FRNODE@$FRUSER \
\ *Sorry, we do not have LISTSERV, try MAILSERV
ECHO     FINFILES .        *        M  DISCARD
ECHO     FINFILES *        *        M  \
\ RUN  /usr/local/bin/send -u echo $FRUSER@$FRNODE *Got Message: '$TEXT'
FINGER   FINFILES .        *        M  DISCARD
FINGER   FINFILES *        *        M  \
\ RUN /usr/local/lib/nje/nje-finger $FRUSER $FRNODE $ARGS

# The default
*        *        *        *        M  BRCAST
```

6.4 Node NJE command 'HELP' response file: `cmd-help.txt`

Node help text can be changed with the `cmd-help.txt`-file, which paths is defined in the `msg-exit.cf`.

```
Commands available for FUNETNJE emulator:
   Help / HILfe / SOS - Three commands to ask for this info
   Query SYStem       - Show line status summary report, and activity
   Query SYStem S     - Same as "Q SYS", but with extra activity data
   Query STATs        - Show line statictics
   Query Nodename     - Show the route entry to that node
   Query linkname A/F - Available via Query SYStem
   Query linkname Q   - Show 30 first files in queue on the link
   CPQuery Names      - List all users logged on
   CPQuery User       - Tell how many users are logged on
   CPQuery User username - Look for a specific username
   CPQuery LOG        - Send the WELCOME message
   CPQuery CPU, CPLEVEL, IND, T - Machine type and time
   MSG node user text.. - A way to send a NMR to some node via msg relay.
   CMD node text..      - A way to send a NMR to some node via cmd relay.
This system has also NMR responding servers at addresses:
   ECHO@FINFILES       -- throws the NMRs back to you.
   MAILSERV@FINFILES   -- ask for HELP
   FINGER@FINFILES     -- Usage: tell finger@finfiles address@internet
```

6.5 BITNET node routing databases: `nje.routes`

FUNETNJE uses *IBM RSCSv1* compatible route files. Used data-elements are *SITE*, and *LINE* definitions (second, and third tokens).

The routing file (for example `finfiles.routes`) is generated with `njeroutes` -command:

```
# njeroutes  finfiles.header finfiles.netinit finfiles.routes
```

Local override file: `finfiles.header`

```
*
*  Header of FUNETNJE/HUyNJE package routing table
*
*  Generic format:
*       SITE    LINE    FORMAT
*
 ROUTE FINFILES LOCAL ASCII
 ROUTE HAMSTER  LOCAL ASCII
 ROUTE FINFTP   LOCAL ASCII
 ROUTE FINUTU   FINUTU EBCDIC
 ROUTE ALIJKU65 ALIJKU65 EBCDIC
 ROUTE ALIJKU64 ALIJKU65 EBCDIC
 ROUTE FINHUTC  FINHUTC EBCDIC
```

Node route file: `finfiles.netinit`

```
*
* GR version 91-11-19, date=94-02-04, time=00:01:34
*
* Table generation parameters: GENROUTE FINFILES NAME=FINFILES.NETI
*                            : NIT.A
*
* Routing table information for
* node: FINFILES nodenum: 0963
*
*
* The routing table neither contains neighbours nor the local node.
* List of neighbours:
*
* node: FIGBOX   nodenum: 3025
*       FINHUTC           1044
*
* * The position of the network name and country *
* * assumes that the RSCS ROUTE command does not *
* * have more than two parameters.               *
ROUTE VERS9402 FINHUTC  EARN     NL
ROUTE FINABO   FIGBOX   EARN     FI
ROUTE FINHUT   FINHUTC  EARN     FI
ROUTE FINHUTA  FINHUTC  EARN     FI
ROUTE FINJYU   FIGBOX   EARN     FI
ROUTE FINKUO   FIGBOX   EARN     FI
ROUTE FINNPHI  FIGBOX   EARN     FI
ROUTE FINOU    FIGBOX   EARN     FI
ROUTE FINUH    FIGBOX   EARN     FI
ROUTE FINUHA   FIGBOX   EARN     FI
...
```

7 Documentation of user programs for NJE use

A set of program for user to handle the files, and communication with the BITNET.

7.1 Query user's reader: `qrdr`

Shows a listing of user's BITNET spool – the 'reader'.

```
QRDR: Spool dir: '/usr/spool/bitspool/MEA/'
From:             To:               FName:   FExt:    Type  Form:    SpoolID
NETSERV@FINHUT    MEA@FINFILES      NETSERV  NOTE     PUN A QUFINHUT 0001
NETSERV@FINHUT    MEA@FINFILES      NETSERV  NOTE     PUN A QUFINHUT 0002
NETSERV@FINHUT    MEA@FINFILES      NETSERV  NOTE     PUN A QUFINHUT 0003
...
```

Synopsis: `qrdr [ -u` *`user`* `] [ -n ] [ -l ] [` *`dirpaths, filepaths`* `]`

qrdr gives users a tool to see what they have in their incoming BITNET spool queues. Depending on file protections, it also lets others to see what user XYZ has in his/her queue, or ANY directory possibly containing BITNET spool files, or just any specified files...

When '-u *user*' has not been used, the default behaviour of *qrdr* is to find out who user is making query by asking 'whoami', eg. getting value of *LOGNAME* environment variable. (On SysV this is *USER* variable.) If the environment doesn't give answer, `/etc/passwd`-file is consulted with effective uid.

Spool directory is determined using same mechanism as in *funetnje* main program. Sample configuration of file procesing exits gives directory `/usr/spool/bitspool` in which UPPER-CASE userids (or pseudoids - MAILER is one such) are 'user' specific subdirectories containing individual spooled files.

Option '-l' gives file paths to spool files, and some heuristics of file contents for *rdrlist(1)* program. For program-parsing usage, the '-l' option uses singular tabulator to separate output fields.

For the '-l' option, recognized data types are: NETDATA, VMSDUMP, CARD (Cornell CARD), and PUNCH when none of the previous ones match to input punch records. For three PRINT formats there are: PRINT (Machine Carriage Control), PASA (ANSI Carriage Control), and PCPDS for rare 3rd alternative.

7.2 Receive files from spool: `receive`

With `receive(1)` user can receive files from the spool with a *spoolid* (a decimal number in range of 1 to 9900), a filename in the user's spool directory, or (relative) path to a file.

Synopsis:

```
receive [-n] [-a|-b] [-asa|-rawpun|-bitraw|-bcat]  [-d]
        [-u username ] [-o outpath ] {filepath|spoolid}
```

Options are as follows:

-n Don't delete the spool file, when processing completes successfully.

-a Do forced translate of the file contents from EBCDIC to ASCII. (Despite of the spool file's class.)

-b Consider the file to be binary, only receive it, don't translate.

-d Debug outputs of the NJE spool file

-asa Affects PRINT files which contain MCC carriage control codes (see bitspool(5)) by translating them to FORTRAN-style carriage control.

-rawpun Outputs the file contents (usually 80 chars per record PUNCH records) as is without adding anything, nor translating it.

-bitraw Something from the earlier versions of these tools; output records in format of <length>+<data,data,data,...>

-bcat Output non-translated records in format:

```
<char of reclen> <chars of record>
```

-u *username*
Look into *username*'s spool directory, instead of who invokes the command.

-o *outpath* Define output file, '-' denotes *STDOUT*.

Input files can be designated either with numeric spoolid, or path(s) to the file(s).

7.3 File sending with `sendfile/print/punch/submit`

This program quintet is actually one binary with four invocation names. It behaves slightly differently with each name:

sendfile/sf Generates *SYSOUT PUNCH* file containing *NETDATA* encoded contents of the original file, and submits it to transmission to a remote system.

print/bitprt
Generates *SYSOUT PRINT* file, and submits it to transmission to a remote system.

punch Generates *SYSOUT PUNCH* file, and submits it to transmission to a remote system.

submit Generates *SYSIN* batch job spool file, and submits it to transmission to a remote batch facility for further processing.

Actual usage details are on the man-page. ****T.B.W. ? ****

7.4 Interactive messages/commands sender: `send`

Synopsis:

```
send [-u fromuser|-s] [-c(ommand)] [@]node [command string]
send [-u fromuser|-s] [-m(essage)] user@node [message text]
```

The first format sends commands to designated node. Using the option '-c' (or '-command') is unnecessary, unless the target node is wanted to be without the '@'-sign on the command line.

The second format sends interactive (*NMR*) messages to the recipient *user@node*. The maximum size of the NMR message is 120 characters per line. Unlike for the files, the network will not guarantee, that the NMR message will reach the destination, however it does a "best effort" attempt at it.

When no command string, or message text is included at the command line, the input is taken from the *STDIN*. If the input comes from somewhere else than TTY, prompting is suppressed, like for example when some command output is piped to `send`.

When *STDIN* is used, message entering ends with the end-of-file condition (Often created with 'ctrl-D' from the keyboard.)

7.5 Send file in multiple parts along with 'table of contents': `bitsend`

This program has been built years back, but not updated

****T.B.W.****

7.6 Receive 'bitsent' files: `bitreceive`

This program has been built years back, but not updated

****T.B.W.****

7.7 Register to the running Emulator to be 'gone': `ygone`

The `ygone`-command instructs the emulator that the user is going away, and that all NMR-messages to him/her should be stored to file `.ygone.message`.

When user returns, and issues command `ygone -d`, the flag is turned off, and all accumulated messages are displayed.

More information on: *man ygone*

7.8 Trap, and pipe out incoming NMR messages from the Emulator: `iucvtrap`

****T.B.W.**** – whole thing needs to be created

7.9 Resend files from user's spool: `transfer`

A special utility for transfering spoolfiles to another destination.

Usage:

```
.../transfer [-t] NEW@DESTIN /absolute/path/to/spool/file
```

This is *very special* program for very rare and special use. Super-user can use **-t**-option to order "transparent" transfer, that is, transfer where original originator is **not** altered.

For normal users the originator address (FRM:) will get rewritten as the caller of this utility.

8 Documentation of system programs for NJE use

These programs are used by system manager, or they are utilities needed for system MTA integration:

8.1 `bmail` – 'sendfile' with a twist to send BSMTP email

A special form of `sendfile` to send email-files to remote systems – to a MTA at there, or (gasp), direct to the user, if no MTA exists at the node...

```
bmail [-u origuser] [-v] [-tag RSCS-TAG]
      [-nd Gateway-Address | -b(smtp) Gateway-address]
      FromUser@FromNode ToUser@ToNode [ToUser@ToNode...]
```

Options are as follows:

-u origuser
: Defines the originating user, e.g. local *MAILER*... Only *super-user*, and *daemon (uid=1)* can use this option.

-v
: Request BSMTP 'VERB ON' to be injected. Sometimes usefull, when debugging some IBM CMS mailers.

-tag RSCS-TAG
: Optional RSCS-TAG string to be passed to the transporter, usually not needed.

-nd Gateway-address
: Send the email in NETDATA format – implies also BSMTP.

-b(smtp) Gateway-address
: Make the file contents to be BSMTP – that is, Batch SMTP. Without NETDATA the width limit is 80 chars.

The message content (along with its RFC-822 headers) is taken from *STDIN*.

8.2 `mailify` – incoming mail-file processing

Normal use for `mailify` is to invoke it from incoming file processing (see `file-exit.cf`):

```
.../mailify /var/spool/.../NJEFILE
```

There are actually two versions of `mailify`, one which is script (`mailify.sh`), and newer C-program, which superceeds the old script.

The `mailify` analyzes the incoming NJE-spool-file, and makes the necessary parsing/transformation of it into RFC-822 format, and feeds it to the system mailer.

In the C-program there is support for `ZMailer`'s zmailer(3) -library, which thus can bypass more generic use of `/usr/lib/sendmail` as a feeder to that mailer.

These incoming mail processors are at their best, when they receive BSMTP in NETDATA file.

8.3 `ucp` – Emulator master control panel

Synopsis:

```
ucp cmd...
  HELP  - Show this message
  SHOW LINE/QUEUE - Show lines or queue status
  START LINE n - Start a closed line
* START STREAM n LINE m - Start specific stream in active line
  SHUT [ABORT] - Shutdown or abort the whole program
```

```
  STOP LINE - Stop a line
  FORCE LINE - Stop a line immediately
* STOP STREAM n LINE m - Stop a single stream in active line
* FORCE STREAM n LINE m - Stop immediately
  QUEUE file-name [SIZE size] - To queue a file to send
  RESCAN EXITS - Rescan file and message exits.
  RESCAN QUEUE - Rescan queue and requeue files.
  RESCAN ROUTE - Reopen route database.
  DEBUG DUMP - Dump all lines buffers to logfile
% DEBUG RESCAN - Rescan queue and requeue files.
  LOGLEVEL n - Set the loglevel to N
  ROUTE xxx TO yyy - Change the routing table.
       Route to  OFF  will delete the route entry.
  GONE username LoginDirectory - Add username to gone list
  UNGONE username - Remove a user from the Gone list.
* - Not yet implemented,  % - obsoleted
```

Program for commanding the FUNETNJE emulator to do various things. Successfull usage of this utility *requires* that the user has supplementary group, which allows communication with the FUNETNJE.

8.4 `namesfilter` – Mailer route extractor from `BITEARN.NODES` file

Synopsis:

```
namesfilter:  Filters BITEARN.NODES data to mailer databases
   namesfilter -zmailer  < bitearn.nodes > bitnet.routes
   namesfilter -sendmail < bitearn.nodes > bitnet.routes
   namesfilter -smail    < bitearn.nodes > bitnet.routes
   namesfilter -netsoft  < bitearn.nodes > bitnet.netsofts
```

This utility parses `BITEARN.NODES`-file, and from it all the '*:type.NJE*' entries are checked up for possible '*:servers*.*'-entries, which define mailers. Of those, only one can exist without "OUTONLY"-tag, and it defines the receiving mailer.

If no non-"OUTONLY" '*:servers*.*' are defined, default mode of "**defrt1**" is used, othervice some smarter version is chosen.

The "-netsoft" dumps `":netsoft."`-tags from `BITEARN.NODES` for the curious (author, of course) to see what software people use.

Originally this utility was written for ZMailer usage, other modes are "afterthoughts"...

To understand more about those tags, see an accompanying file `newtags.descript` for more info.

8.5 `njeroutes` – Route-database generator program

Synopsis:

```
njeroutes  header.file routes.file output.file
```

This program generates local *BINTREE* copy of the route database from couple input files. Runtime system uses the produced file to look up the NJE routes.

8.6 `netinit.sh` – Automatic route-updater

`netinit.sh` is a small script which can be run from `cron` to facilitate automatic system maintance.

The sample version of it queries pseudo-user `NETINIT`s reader and if it find there `FINFILES.NETINIT`, or `BITEARN.NODES`, it processes them for routes (NETINIT), and ZMailer BITNET email routing database (ROUTES).

Sample output below (May-1995 route-db)

```
FINFILES.NETINIT: .../bitspool/NETINIT/0001 NETDATA NETSERV@FINHUTC   \
                                        NETINIT@FINFILES FINFILES.NETINIT N
Ndparse ok, generating routes..
Records read:
Processing routing table
100 200 300 400 500 600 700 800 900 1000
1100 1200 1300 1400 1500 1600 1700 1800 1900 2000
2100
Total records inserted: 2136, 0 duplicates in DUPLICATE.TMP
```

8.7 `acctcat` – Utility to dump the RSCS-accounting records

Synopsis:

```
acctcat /path/to/rscs/acct-file
```

UNIX-program that is able to dump VM/CMS RSCS-accounting records into ASCII format. (For various reasons that format was chosen as the accounting format of the FUNETNJE as well.)

See file `rscsacct.format` for the log-file format. The ASCII-dump via `acctcat` is almoast identical.

9 Interfacing ZMailer/smail/sendmail with FUNETNJE

Here are some instructions on how to interface various MTA-systems into the *FUNETNJE* package.

There is a separate program (*mailify*) responsible for incoming email processing, these are routers, and senders of BITNET email (*bmail*), and there is route database generator for generating the necessary *nodename->route* mapping (*namesfilter*).

9.1 Interfacing Toronto/FUNET ZMailer with FUNETNJE

Doing this interface was easy due to vast amounts of easy to understand configurability of the *ZMailer*[1] system.

Depending on if the system is to be a smart-mailer, or not, ZMailer's `router.cf` shall contain transport preference definition:

```
# The transport preference order
protocols='routes smtp usenet bitnet'
```

which instructs it to include those four protocols into the route-analysis. The protocols are attempted in given order! Host BITNET name must be defined in `/etc/name.bitnet` for ZMailer's BITNET-protocol.

If it is desirable **not to** have smart BITNET routing at the system, it is possible to use lazy-routing of all "*.BITNET"-addresses into one system somewhere else with following definition in `/etc/mail.conf`:

```
...
# rrouter.cf uses these..  Uncomment for use
# : ${BITNETACTION:="bsmtp3nd mailer@searn"}
# : ${BITNETACTION:="smtp figport.funet.fi"}
```

the first one uses special delivery via BITNET facilities to MAILER@SEARN, the second uses ordinary SMTP.

BITNET email routing database (`$MAILVAR/db/routes.bitnet`) contains entries like:

```
academic        bsmtp3!mailer!academic
acmvm           bsmtp3!mailer!acmvm
acusd           defrt1!acusd
...
searn           bsmtp3nd!mailer!searn
...
```

which essentially mean:

academic Use **bsmtp3**-channel, and send it to **mailer@academic**

acusd Use **defrt1**-channel, and send mail do **$user@acusd**

searn Use **bsmtp3nd**-channel (bsmtp3 in NETDATA instead of PUNCH), and send it to **mailer@searn**

That email routing database is built from `BITEARN NODES`-file by use of `namesfilter` -program:

```
# ./namesfilter -zmailer < bitearn.nodes | sort > routes.bitnet
```

[1] ZMailer is a mailer running in two modes with multiple programs doing their parts; `router` (1 to many) doing email address routing, `scheduler` running transport channels, like `smtp`, `mailbox` (local delivery), etc. There is also incoming channel program `smtpserver`, and compability interface program `/usr/lib/sendmail`.

The ZMailer is available from ftp://ftp.funet.fi/pub/unix/mail/zmailer/zmailer-*vers-datecode*.tar.gz, via anonymous FTP.

Unlike the original Toronto `namesfilter`-package, this does not take `BITEARN DOMAINS`, or `XMAILER NAMES`.

Channels are defined in `$MAILSHARE/scheduler.conf`, in which these five are defined as:

```
...
# The default definition
*/*
        ...
        # default uid/gid of transport agents
        user=root
        group=daemon
...
defrt1/*
        command="sm -c $channel defrt1"
bsmtp3/*
        command="sm -c $channel bsmtp3"
bsmtp3rfc/*
        command="sm -c $channel bsmtp3rfc"
bsmtp3nd/*
        command="sm -c $channel bsmtp3nd"
bsmtp3ndrfc/*
        command="sm -c $channel bsmtp3ndrfc"
```

Meaning they all are run via 'sendmail-compatible transporter'.

Configuration of the 'sendmail-compatible transporter' at `$MAILSHARE/sm.conf` contains these five entries for the BITNET:

```
...
#
# bitnet stuff F=hu not set
#
bsmtp3      snmSX /usr/local/lib/nje/bmail bmail -b $h -u MAILER $g $u
bsmtp3rfc   snmSX /usr/local/lib/nje/bmail bmail -b $h -u MAILER $g $u
bsmtp3nd    snmSX /usr/local/lib/nje/bmail bmail -nd $h -u MAILER $g $u
bsmtp3ndrfc snmSX /usr/local/lib/nje/bmail bmail -nd $h -u MAILER $g $u
defrt1      snS   /usr/local/lib/nje/bmail bmail $g $u
```

9.2 Interfacing Smail3.0 with FUNETNJE

This information is from Gerald Hanusch <root@alijku65.edvz.uni-linz.ac.at>

Relevant files are on subdirectory `smail-configs/` *of the source directory.*

In order to get SMAIL3.1.28 working with the FUNETNJE emulator, following steps are necessary:

1. Copy the BSMTP3, BSMTP3ND and DEFRT1 BASH scripts to your *SMAIL-LIBDIR*, e.g. `/usr/local/lib/smail/`
2. Edit these 3 Scripts to fit your nje emulator installation, mainly the path to the `bmail` program coming with the emulator needs to be set properly.
3. If you don't use a `routers` and `transports` file till now, because the compiled in defaults of SMAIL fit your needs, make a copy of these files to the *SMAIL-LIBDIR*. The source distribution of SMAIL comes with generic samples, which reflect the compiled-in configuration.
4. Make an entry to `routers` file like below. Commonly between the entry which defines the `path` router and the `uucp-neighbors` router with the following lines

```
bitnet: driver=pathalias,
        method=bitmethods;
        file=bitearn,
        proto=bsearch,
        -optional,
        -required,
        domain=bitnet,
```

The position of the entries within `routers` file is significant. The decision, which path is taken, depends mainly on this position. Please consult the SMAIL doc, resp. man 5 smail for more info on this.

5. Make a directory `methods` within *SMAIL-LIBDIR*. Copy the supplied `bitmethods` file into there.

   Edit or generate a bitnet mailers routing table, and copy it to *SMAIL-LIBDIR*. It can be built from `BITEARN NODES`-file by use of `namesfilter`-program:

   ```
   # ./namesfilter -smail < bitearn.nodes | sort > routes.bitnet
   ```

   The table has the common pathalias format with following entries:

   ```
   destdomain   transportmethod!mailername!nodename!%s
   ```

   where *destdomain* is the domain within bitnet, *transportmethod* is either bsmtp3, bsmtp3nd or defrt1, *mailername* is the name of the remote mailer userid which receives email from bitnet, and *nodename* is the bitnet node running the mailer

   The '*%s* behind the *nodename* is not needed for our purposes, but is necessary to pass the syntax check. The meaning of the 3 transport methods is described in `bmail(8)` utility.

6. Edit the `transports` file in *SMAIL-LIBDIR* by adding the entries bstmp3, bsmtp3nd, and defrt1. A sample is provided with the nje emulator package.
7. Add to your SMAIL - `config` file to the *domains=*-line an entry for *bitnet*, separated by a ":"
8. If the packages installed properly, reload smail configuration. Now it should be possible to deliver mail within BITNET using nje.

9.3 Interfacing BSD-sendmail with FUNETNJE

This presentation is by the courtesy of mr. Jukka Ukkonen of CSC.FI, translation to english is by Matti Aarnio

See source-tree subdirectory `sendmail-configs/` *for the way how Paul Bryant* <*P.Bryant@rl.ac.uk*> *did this.*

Note also from ZMailer's bsmtp3 et.al. entry definitions how you can change the local "MAILER"-name. (The "-u"-option for the bmail-program.)*

Somewhere in the begin of the `sendmail.cf`:

```
# Bitnet route table (overriding default routes)
Kbitnettable dbm -o /usr/local/lib/mail/bitnettable
```

Then in *S0*-rules after error-mailer, local checkup, uucp, and mailertable:

```
# Try a separate table for bitnet routes.
R$*<@$-.$+.bitnet.>$*       $:<$(bitnettable .$3$)>$1<$2.$3>$4
R$*<@$+.bitnet.>$*          $:<$(bitnettable $2$)>$1<$2>$3
R<$-!$-!$+>$*<$+>$*         $#$1$@$3$:$2          matched a 3 part route
R<$-!$+>$*<$+>$*            $#$1$@$2$:$3@$4       matched a 2 part route
```

One completely new ruleset, though a short one, because other domains need not stripping:

```
#
```

```
#  Strip fake domains off in case posting directly
#  to the receipient's native mail system.
#
S22
R$*<@$+.bitnet.>       $@$1<@$2>
```

This *R=22* is only for receiver's side mailer definitions. (Odd tens are for the sender, even tens are for the receiver. Smaller ten is for envelope conversion, and bigger is for header conversion in case headers call for different transformation, than envelope, a'la S=11/31.)

If sender address need to be translated into node format (from their propable FQDN format), there appears a need for yet another mapping of FQDN addresses to BITNET node names, which in practice means yet another dbm-database for maintance:

```
Kbitnetname dbm -o /usr/local/lib/mail/bitnetname
```

This needs another set like '22' above which tries to do the lookup according to the sender's machine, othervice the default *S=11* is ok.

On the other hand those, who have *S=11/31*, like I have done it, the '31' is dangerous, because it does a lookup of '*$(generics $1 $)*', which produces long names a'la *Jukka.Ukkonen*, which in the usual BITNET-world is not quite commonly tolerated. . .

A new *S=xx* could be for example this S=12 below. '**$N**' can be made local default name for the BITNET; **CNfinfiles**:

```
S12
R$*<@$=w.>     $:$1<@$(bitnetname $2$)>
R$*<@$=w.>     $@$1<@$N>
R$*<@$+.>      $@$1<@$(bitnetname $2$)>
```

In addition to those there is also a need for all the mailer entries, like: '**Mbsmtp3**' (See ZMailer's "sm"-defines for hints.)

Non-translated email continues with an apology of being too tired to finish it "tonight", nor is 'Jau' certain, that these make any sense/are corrent – early September-94.

```
          Cheers,
                  // jau
------
  /    Jukka A. Ukkonen, M.Sc. (tech.) Centre for Scientific Computing
 /__   Internet: ukkonen@csc.fi            Tel:  (Home) +358-0-578628
   /   Internet: jau@cs.tut.fi                    (Work) +358-0-4573208
  v    X.400:    c=fi, admd=fumail, no prmd, org=csc, pn=jukka.ukkonen
```

10 Technical documents about insides of things

10.1 Directory structure of the FUNETNJE

`/etc/funetnje.cf`
: Is the default location of the runtime system master configuration file. It defines all other files and directories used by this system.

`/usr/local/funetnje/`
: A possible location to place system programs, which are used by the system only.

`/usr/local/bin/`
: A possible location to place user-callable programs into.

`/usr/local/sbin/`
: A possible location to place sysmanager-callable programs into.

`/var/spool/bitnet/`
: Outgoing file spooling directory (and its subdirectories); defined on `/etc/funetnje.cf`.

`/var/spool/bitspool/`
: Incoming file spooling directory – each user has his/her own subdir; defined on `/etc/funetnje.cf`.

10.2 Client to Emulator messaging

There are multiple methods (depending upon your operating system) on how to use the client->emulator command channel.

The coded alternatives are:

UDP Propably the best alternative. System allows UDP-datagram sending to just anybody, but not all can access (from filesystem) the necessary magic value to present authorization.

Socket IP address is taken from `/etc/funetnje.cf` *CMDMAILBOX* entry (numeric), and it naturally must be on same machine, as the emulator runs. (Client reads the supposed peer address from that location.)

Example:

```
CMDMAILBOX  UDP 127.0.0.1 175
```

FIFO Uses named pipe for IPC – not outmost reliable method, and can't grant universally working the same way on every machine.

Fifo path is taken from `/etc/funetnje.cf` *CMDMAILBOX* entry.

Example:

```
CMDMAILBOX  FIFO /var/spool/bitnet/.cmdfifo
```

SOCKET An `AF_UNIX`/`SOCK_STREAM` socket; sort of named pipe on BSD world. Also somewhat system dependent.

Socket path is taken from `/etc/funetnje.cf` *CMDMAILBOX* entry.

Example:

```
CMDMAILBOX  SOCKET /var/spool/bitnet/.cmdsocket
```

Only the first character of the type identifier (UDP/FIFO/SOCKET) is used, and the scan proceeds over the token, and to the next argument.

All methods will need a 4-byte magic token with which they will prefix each communicated message with the server. If the token does not match, the server will throw away the message, and be silent.

The magic token is stored on `/QUEUEDIR/.socket.key`-file.

Such arrangement because there is no universally trustable method of limiting the communication in between processes. Therefore the next-best is to use filesystem semantics to limit access to ordinary files containing magic tokens.

10.3 IUCVTRAP <=> Emulator messaging

****T.B.W.**** – whole thing needs to be created

10.4 FUNETNJE Spool Format

Straight copy of man-page: bitspool(5)

Bitspool format consists of an **ASCII** header, and a binary body containing length-tagged "records". The length-tag is "*network byte order*"-ordered 16 bit short integer. This is to facilitate transfer of spool files in between systems, and to make it simpler to use non-homogenous hardware in clusters which share the same **BITNET** node..

Known **ASCII** headers are:

FRM: USERNAME@HOSTNAME
: Mandatory (fall-back default: "***@****"). Uppercase-only data in fixed 17 chars wide field, left justified.

TOA: USERNAME@HOSTNAME
: Mandatory (fall-back default: "POSTMAST@<localname>"). Uppercase-only data in fixed 17 chars wide field, left justified.

FNM: FILENAME
: Optional, default value: "UNKNOWN". Mixed-case alphanumeric data in fixed 12 chars wide field, left justified.

EXT: FILEEXTN
: Optional, default value: "DATA". Mixed-case alphanumeric data in fixed 12 chars wide field, left justified.

TYP: type
: Optional, defaults to "PUNCH". Controls physical device attribute on the outgoing NJE headers. When sending PRINT files, use **TYP: PRINT**. When you want to send SYSIN jobs, use **TYP: JOB**, and if it is just a PUNCH something, use **TYP: PUNCH**.

  Recognized obsolete keywords for the TYP are: MAIL, BSMTP, FILE, and FASA, which are in reality synonyms for PUNCH, and PASA, which is synonym for PRINT.

CLS: class
: Optional, defaulted by the **TYP:**, if not defined. This is **one** (1) uppercase alpha in range A-Z.

  Customary values are: MAIL, BSMTP files: M, NETDATA containing binary data: N, console printouts: O, all others: A.

FOR: FILEFORM
: Optional, recommended. If you don't want to see file-traversal notes from the network, BITNET style is to start the form name with two characters: "QU". That matches at "QUIET", "QUMAIL", ...

  System wide default is taken from `/etc/funetnje.cf DEFFORM` entry, and it often has value **STANDARD**.

FMT: spool-format
Mandatory (fall-back default: **BINARY**).

BINARY for all locally generated files, and for cases where the **ASCII** headers have been mungled after the file was received from the network.

EBCDIC denotes spool file which has been received from the network, and contains all possible data blocks in it. (Users will ignore all, but PUNCH, and PRINT records..)

FID: 9900 Mandatory, sender original spoolid; A number between 1, and 9900. File submitter should put "0000" in there, and delivery facility will do something about it.. When system picks up this field, it will assign there next spoolid from file sender's store, however network delivery uses "**OID:**", which it assigns..

OID: 0000 Mandatory, for spool-id allocation at the first time it is queued for delivery to somewhere remote.

TAG: RSCS-TAG-STRING
Optional. The IBM NJE TAG string of 136 characters.

DIS: RSCS-DIST
Optional. Fixed width, left justified 8 characters field of RSCS DIST information. Default: "**SYSTEM**".

REC: nnnn
Highly preferred (almoast mandatory!) Tells the number of records of data; necessary to get job-header right on file transmission, altough not really mandatory for successfull file sending. As the count of records isn't known before all are written (in the usual case), this entry is written at first to reserve space for later seek, and rewite with correct information.

JNM: JOBNAME
Optional. Contains NJE job name (8 characters).

PRT: prtfiles@tonode
Optional. For **SYSIN** jobs this defines alternate address for sending the job PRINT outputs to. Default is taken from "**FRM:**".

PUN: punfiles@tonode
Optional. For **SYSIN** jobs this defines alternate address for sending the job PUNCH outputs to. Default is taken from "**FRM:**".

END: Mandatory. Ends the **ASCII**-portion of the headers. Terminating newline must be immediately after the colon character.

With all (except **END:**) above header tags there must be exactly one space (ASCII 32) after the colon, and before data. With the **END:**, there must not be any extra blanks after the colon, and before terminating newline.

Order of the **ASCII** format headers is not fixed, except that **END:** must be last.

Binary format data follows syntax:

```
<16-bit length> <length number of bytes>
```

where the **<16-bit length>** is encoded as **network-byte-order** short integer. This is to facilitate easy copying of *funetnje*(8) spool files from one machine to another. This also makes it possible to share common **BITNET** pool between several machines independent of their native byte order.

Sample data formats:

PUNCH

```
(any content, like plain text, and  NETDATA)
```

```
0x80, 80,                          0x80 = PUNCH, 80=content length
<80 bytes of EBCDIC/whatever>      The PUNCH content, 80 bytes
```

PRINT Machine Carriage Control:

```
0x90, 132, MCCbyte,                0x90 = PRINT MCC, 132=content length
<132 bytes of EBCDIC/whatever>     The PRINT content, 132 bytes
```

PRINT ASA

Carriage Control:

```
0xA0, 132, ASAbyte,                0xA0 = PRINT ASA, 132=content length
<132 bytes of EBCDIC/whatever>     The PRINT content, 132 bytes
```

PRINT CPDS

```
Carriage Control:
0xB0, 132, CPDSbyte,               0xB0 = PRINT CPDS, 132=content length
<132 bytes of EBCDIC/whatever>     The PRINT content, 132 bytes
```

Trailing **EBCDIC**-spaces can be suppressed in the spool files, thus above are given ranges of lengths. If the record content is shorter, than its defined size, receiver *must* pad it with **EBCDIC**-spaces. (Of course, this way spool-size of a megabyte of spaces will be 13108 records of (2+)2 bytes + header, which makes circa 53kB. When received, it will expand by a fractionally filled last record.) The length-byte at each record *must* be of its nominal value, record truncation is coded with the 16-bit-length value.

The *MCCbyte*, and *ASAbyte* are two kinds of "*carriage control*" -mechanisms. The *CPDSbyte* is third kind, however documenting it is left for further study.

IBM printers are fixed 132 characters wide, and on overall those systems function on records, and a write in this respect means write of whole line. On overall the IO-operations on IBM mainframe happen on records, or preferrably chains of records on which single start-IO instruction can be used to start the IO-channel.

Carriage-control mechanisms contain two kinds of controls, both pre-write, and post-write controls exist in MCC mode. In ASA-mode only pre-writes are defined. Pre-write controls happen before the printout line is written, for example 'jump to nearest VFU channel 1 forward' (in plain english: Form-Feed..)

Jump to next page
ASA: '1', Pre-MCC: X'8B'

Overwrite; Defined only in pre-write.
ASA: '\+', MCC: X'01'

Write and advance one line
ASA: Space, Pre-MCC: X'0B', Post-MCC: X'09'

Write and advance twolines
ASA: '0', Pre-MCC: X'13', Post-MCC: X'11'

Write and advance three lines
ASA: '-', Pre-MCC: X'1B', Post-MCC: X'19'

Other formats, than 0x80, 0x90, 0xA0, 0xB0 of binary portion records should be ignored. (Received network transport layer headers are there, etc.)

10.5 Implemented NETDATA features

The *parser* understands following NETDATA keys:

•INMDSORG
Only SEQUENTIAL is understood (others generate complaint, but do not abort.)

•INMRECFM
No special record-contents are supported, nor flagged!
Carriage-control information is checked against MCC.
Blocking info is ignored.
Record-kind info is ignored.

•INMLRECL
If LRECL exceeds 65534, it is complained about, but not aborted.

•INMSIZE
SIZE (in bytes) is ignored.

•INMNUMF
NUMF (what??) is ignored.

•INMFACK
Request for ACK of reception is analyzed, and provided for user for possible action.

•INMTERM
This flag of PROFS NOTE message is noted, but ignored.

•INMFTIME
This file's last modification timestamp is analyzed, and provided for user for possible action.

•INMCREAT
This file creation timestamp is analyzed, and provided for user for possible action.

•INMDSNAM
The originating system dataset name (esp. on MVS systems) is saved for possible user action.

•INMMEMBR
The originating system dataset member name ("file name") is saved for possible user action.

•All others are silently ignored

NETDATA producing systems (`sendfile`-command mainly) create following datas:

On INMR01 -record:

•INMFUID
Sending user's username is stored into the file. Actually this *should* be the owner name of the sent file.

•INMFNODE
Sending system name is stored into the file. (In theory this should be file's originating node name.)

•INMTUID
File senders username

•INMTNODE
File senders nodename

•INMFTIME
If the sent file is a regular file, this is its last-modify timestamp. Otherwice a 0-size date is stored (so the INMFTIME key is stored, but with void data.)

- INMLRECL
  On INMR01-record the LRECL is fixed 80.
- INMNUMF
  Fixed value of "1".
- INMFACK
  If an acknowledge is wanted, this key is generated.

On INMR02 -record:

- INMUTILN
  A fixed value of "INMCOPY" (7 chars) is stored.
- INMDSORG
  A fixed value of "PHYSICAL" (0x4000) is stored.
- INMLRECL
  A *true* LRECL (user supplied) is stored onto INMR02-record.
- INMRECFM
  A true RECFM (user supplied) is stored onto INMR02-record.
- INMSIZE
  File size in bytes is stored.
- INMDSNAM
  Here a dataset name is stored in style of: "A Filename Fileext"
  The first letter is always "A", and then comes one or two at most 8-chars long tokens. One space separates each token.
- INMCREAT
  If file creation time is supplied, it is stored.

On INMR03-record:

INMRECFM
Fixed value of 0x0001.

INMLRECL
Fixed value of 80.

INMDSORG
Fixed value of 0x4000.

INMSIZE Sent file's size in bytes.

10.6 Emulator logics

Documenting how the emulator works internally...

****T.B.W.****

11 Literature

Here is a partial list of things to read:

- IBM publication SC23-0070-02: *Network Job Entry – Formats and Protocols*
- IBM publication GG22-9373-02 *(older version of previous)*
- `NEWTAGS DESCRIPT` from your nearest NETSERV.
- `NETDATA DESCRIPT` from your nearest NETSERV.
- `BRFC0002 TEXT` from BITNIC
- `TECOVER.LISTPS` from Princeton – BITNET-II technical overview. (LISTSERV@PUCC ??)
- `USRGUIDE.LISTPS` from Princeton – VMNET users/operators guide. (LISTSERV@PUCC ??)

Table of Contents