FILE TRANSFER PROTOCOL

Meeting Announcement and a New Proposed Document

Attached is a new proposal for a File Transfer Protocol. The
document is an extensive update to RFC 354 and, I believe,
incorporates solutions to most of the objections to RFC 354.

It now seems appropriate to make another attempt to reach final
agreement on FTP. Accordingly, I am calling a meeting of interested
parties, to be held at BBN on March 16, for discussion of this and
other proposals.

This note is directed to the network community at large, rather than
specifically to the old FTP committee, because I don't believe that
the FTP committee membership includes all the individuals who have
contributed to the current state of FTP design. Nevertheless, it is
intended that the meeting proceed from the current state, rather than
bringing new members up-to-speed. Prospective attendees should
therefore be familiar with at least the following documents:

    RFC 354
    RFC 385
    RFC 414
    RFC 418
    RFC 438

Anyone wishing to attend this meeting should contact Alex McKenzie
(NIC Ident aam) at BBN, 50 Moulton Street, Cambridge, Mass. 02138.
My telephone number is:

                    (617) 491-1850 ext.441

When there is some indication of the number of individuals planning
to attend, a meeting room will be reserved and more specific
information will be directed to attendees.

PROPOSED FILE TRANSFER PROTOCOL

This document is the outcome of a meeting held 25 January 1973 in
Cambridge, Massachusetts, by the following people:

    Abhay Bhushan (MIT - DMCG)

    Bob Bressler (BBN - NET)

    Bob Clements (BBN - TENEX)

    Alex McKenzie (BBN - NET)

    Nancy Neigus (BBN - NET)

    Ken Pogran (MIT - MULTICS)

    Marc Seriff (MIT - DMCG)

The basis of the document is RFC 354 with considerations drawn from
RFC's 385, 414, 418, and 438 and personal communication with network
participants.

                        PROPOSED FILE TRANSFER PROTOCOL

INTRODUCTION

    The File Transfer Protocol (FTP) is a protocol for file transfer
    between HOSTs (including terminal IMPs), on the ARPA Computer Network
    (ARPANET). The primary function of FTP is to transfer files
    efficiently and reliably among HOSTs and to allow the convenient use
    of remote file storage capabilities.

    The objectives of FTP are 1) to promote sharing of files (computer
    programs and/or data), 2) to encourage indirect or implicit (via
    programs) use of remote computers, 3) to shield a user from
    variations in file storage systems among HOSTs, and 4) to transfer
    data reliably and efficiently. FTP, though usable directly by a user
    at a terminal, is designed mainly for use by programs.

    The attempt in this specification is to satisfy the diverse needs of
    users of maxi-HOSTs, mini-HOSTs, TIPs, and the Datacomputer, with a
    simple, elegant, and easily implemented protocol design.

    This paper assumes knowledge of the following protocols:

        1) The HOST-HOST Protocol (NIC #8246)

        2) The Initial Connection Protocol (NIC #7101)

        3) The TELNET Protocol (NWG/RFC #318, NIC #9348)

II. DISCUSSION

    In this section, the terminology and the FTP model are discussed.
    The terms defined in this section are only those that have special
    significance in FTP.

II.A Terminology

    ASCII   The USASCII character set as defined in NIC
                        #7104. In FTP, ASCII characters are defined to
                        be the lower half of an eight bit code set (i.e.,
                        the most significant bit is zero).

    access controls   Access controls define users' access privileges
                        to the use of a system, and to the files in that
                        system. Access controls are necessary to prevent
                        unauthorized or accidental use of files. It is
                        the prerogative of a server-FTP process to
                        provide access controls.

byte size   The byte size specified for the transfer od data.
                     The data connection is opened with this byte
                     size. Data connection byte size is not
                     necessarily the byte size in which data is to be
                     stored in a system, and may not be related to the
                     structure of data.

data connection   A simplex connection over which data is
                     transferred, in a specified byte size, mode and
                     type. The data transferred may be a part of a
                     file, an entire file or a number of files. The
                     data connection may be in either direction
                     (server-to-user or user-to-server).

data socket   The socket on which a User-FTP process "listens"
                     for a data connection.

EOF   The end-of-file condition that defines the end of
                     a file being transferred.

EOR   The end-of-record condition that defines the end
                     of a record being transferred.

error recovery   A procedure that allows a user to recover from
                     certain errors such as failure of either HOST
                     system or transfer process. In FTP, error
                     recovery may involve restarting a file transfer
                     at a given checkpoint.

FTP commands   A set of commands that comprise the control
                     information flowing from the user-FTP to the
                     server-FTP process.

file   An ordered set of computer data (including
                     programs) of arbitrary length uniquely identified
                     by a pathname.

mode   The mode in which data is to be transferred via
                     the data connection. The mode defines the data
                     format including EOR and EOF. The transfer modes
                     defined in FTP are described in Section III.C.

NVT   The Network Virtual Terminal as defined in the
                     ARPANET TELNET Protocol.

NVFS   The Network Virtual File System. A concept which
                    defines a standard network file system with
                    standard commands and pathname conventions. FTP
                    only partially embraces the NVFS concept at this
                    time.

pathname   Pathname is defined to be the character string
                    which must be input to a file system by a user in
                    order to identify a file. Pathname normally
                    contains device and/or directory names, and file
                    name specification. FTP does not yet specify a
                    standard pathname convention. Each user must
                    follow the file naming conventions of the file
                    systems he wishes to use.

record   A sequential file may be structured as a number
                    of contiguous parts called records. Record
                    structures are supported by FTP but are not
                    mandatory.

reply   A reply is an acknowledgement (positive or
                    negative) sent from server to user via the TELNET
                    connections in response to FTP commands. The
                    general form of a reply is a completion code
                    (including error codes) followed by an ASCII text
                    string. The codes are for use by programs and
                    the text is for human users.

server-FTP process A process or set of processes which perform the
                    function of file transfer in cooperation with a
                    user-FTP process. The server-FTP process must
                    interpret and respond to user commands and
                    initiate the data connection.

server site   A HOST site which has a server-FTP process.

server-TELNET   A TELNET process which listens on a specified
                    socket for an ICP initiated by a user-TELNET, and
                    performs in accordance with the ARPANET TELNET
                    Protocol.

TELNET connections The full-duplex communication path between a
                    user-TELNET and a server-TELNET. The TELNET
                    connections are established via the standard
                    ARPANET Initial Connection Protocol (ICP).

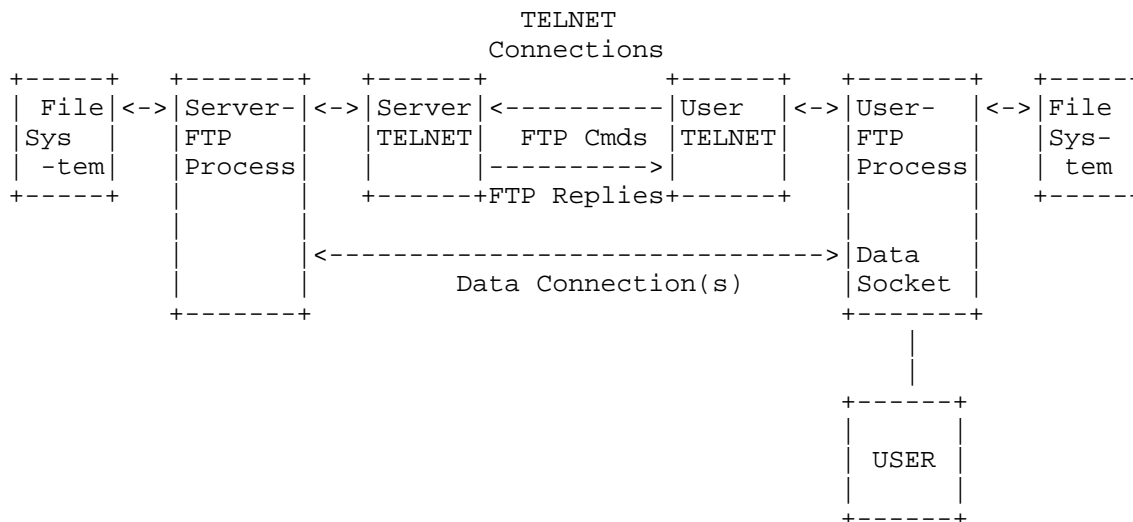type   The data representation type used for data
                        transfer and storage. Type implies certain
                        transformations between the time of data storage
                        and data transfer. The representation types
                        defined in FTP are described in Section III.B.

user   A process on behalf of a human being or a human
                        being wishing to obtain file transfer service.

user site   A HOST site satisfying any of the following
                        conditions: 1) The site where a user is located,
                        2) a site where a user-FTP process is located, 3)
                        a site to which a data connection is made by a
                        server. In the normal case, the sites defined by
                        1, 2, and 3 are the same site, but nothing in FTP
                        requires that this be so.

user-FTP process   A process or set of processes which perform the
                        function of file transfer in cooperation with a
                        server-FTP process. The user-FTP process 1)
                        initiates the ICP (via a user-TELNET), 2)
                        initiates FTP commands and 3) "listens" on the
                        data socket for the data connection. In some
                        obvious cases (use from TIPs and other mini-
                        HOSTs) a user-FTP process will be subsumed under
                        the term "user".

user-TELNET   A TELNET process which initiates an ICP to a
                        specified server-TELNET socket, and performs in
                        accordance with the ARPANET TELNET protocol.

II.B The FTP Model

   With the above definitions in mind, the following model (shown in
   Figure 1) may be diagramed for an FTP service.

   In the model described in Figure 1, the user-TELNET initiates the
   TELNET connections. Standard FTP commands are then generated by the
   user and transmitted to the server site via the TELNET connections.
   FTP commands are in ASCII, in accordance with NVT conventions and the
   TELNET protocol. Note that commands may be initiated by the user
   directly through the user-TELNET or via a user-FTP process. Standard
   replies are sent from the server to the user in response to the
   commands over the TELNET connections.

The FTP commands specify the parameters for the data connection (data
socket, byte size, transfer mode, representation type, and format)
and the nature of file system operation (store, retrieve, append,
delete, etc.). The user-FTP process or its designate should "listen"
on the specified data socket, and it is the server's responsibility
to initiate the data connection and data transfer in accordance with
the specified data connection parameters. It should be noted that
the data socket need not be in the same HOST that initiates the FTP
commands via the TELNET connections, but the user or his user-FTP
process must ensure a "listen" on the specified data socket. A
practical example of such file transfer to third HOSTs is a maxi-HOST
user (who may actually be a TIP user) wishing to transmit a file to
or from an I/O device attached to a TIP. It should also be noted
that two data connections, one for send and the other for receive,
may exist simultaneously.

```
                               TELNET
                             Connections
+-----+   +-------+   +------+              +------+   +-------+   +-----+
| File|<->|Server-|<->|Server|<----------|User   |<->|User-  |<->|File |
|Sys  |   |FTP    |   |TELNET|  FTP Cmds |TELNET |   |FTP    |   |Sys- |
| -tem|   |Process|   |      |  |---------->|      |   |Process|   | tem |
+-----+   |       |   +------+FTP Replies+------+   |       |   +-----+
          |       |   |                              |       |
          |       |   |<----------------------------->|Data   |
          |       |   |       Data Connection(s)     |Socket |
          +-------+                                   +-------+
                                                          |
                                                          |
                                                     +------+
                                                     |      |
                                                     | USER |
                                                     |      |
                                                     +------+
```

  Notes: 1. The data connection may be in either direction.

          2. The data connection need not exist all of the time.

          3. The distinctions between user-FTP and user-TELNET, and
             between server-FTP and server-TELNET may not be as
             clear-cut as shown above. For example, a user-TELNET may
             be directly driven by the user.

             FIGURE 1 Model for FTP Use

The protocol requires that the TELNET connections be open while data
transfer is in progress. It is the responsibility of the user to
close the TELNET connections when finished using the FTP service.
The server may abort data transfer if the TELNET connections are
closed.

III.   DATA TRANSFER FUNCTIONS

Data and files are transferred only via the data connection. The
transfer of data is governed by FTP data transfer commands received
on the TELNET connections. The data transfer functions include
establishing the data connection to the specified data socket in the
specified HOST (using the specified byte size), transmitting and
receiving data in the specified representation type and transfer
mode, handling EOR and EOF conditions, and error recovery (where
applicable).

III.A Establishing Data Connection

The user site shall "listen" on the specified data socket, prior to
sending a transfer request command. The FTP request command
determines the direction of data transfer, and the socket number (odd
or even) which is to be used in establishing the data connection.
The server on receiving the appropriate store or retrieve request
shall initiate the data connection to the specified user data socket
in the specified byte size (default byte size is 8 bits), and send a
reply indicating that file transfer may proceed. Prior to this
reply, the server should send a reply indicating the server socket
for the data connection. The user may use this server socket
information to ensure the security of his data transfer. The server
may send this reply either before or after initiating the data
connection.

The byte size for the data connection is specified by the BYTE
command. It is not required by the protocol that servers accept all
possible byte sizes. The use of various byte sizes is for efficiency
in data transfer and servers may implement only those byte sizes for
which their data transfer is efficient. It is, however, required
that servers implement at least the byte size of 8 bits.

After the data transfer is completed, it is the server's
responsibility to close the data connection, except when the user is
sending the data. In stream mode the sender must close the data
connection to indicate EOF, i.e., completion of the transfer.
Closing the connection is a server option except under the following
conditions:

1)   The server receives an abort command from the user.

2)   The socket or the byte size specification is changed by the
          user.

3)   The TELNET connections are closed.

4)   An irrecoverable error condition occurs.

It should be noted that if none of the above conditions occur it is
possible to maintain two simultaneous data connections, for send and
receive.

III.B Data Representation and Storage

Data is transferred from a storage device in sending HOST to a
storage device in receiving HOST. Often it is necessary to perform
certain transformations on the data because data storage representa-
tions in the two systems are different. For example, NVT-ASCII has
different data storage representations in different systems. PDP-10'
s generally store NVT-ASCII as five 7-bit ASCII characters, left-
justified in a 36-bit word. 360's store NVT-ASCII as 8-bit EBCDIC
codes. Multics stores NVT-ASCII as four 9-bit characters in a 36-bit
word. It may be desirable to convert characters into the standard
NVT-ASCII representation when transmitting text between dissimilar
systems. The sending and receiving sites would have to perform the
necessary transformations between the standard representation and
their internal representations.

A different problem in representation arises when transmitting binary
data (not character codes) between HOST systems with different word
lengths. It is not always clear how the sender should send data, and
the receiver store it. For example, when transmitting 32-bit bytes
from a 32-bit word-length system to a 36-bit word-length system, it
may be desirable (for reasons of efficiency and usefulness) to store
the 32-bit bytes right-justified in a 36-bit word in the latter sys-
tem. In any case, the user should have the option of specifying data
representation and transformation functions. It should be noted that
FTP provides for very limited data type representations. Transforma-
tions desired beyond this limited capability should be performed by
the user directly or via the use of the Data Reconfiguration (DRS,
RFC #138, NIC #6715). Additional representation types may be defined
later if there is a demonstrable need.

Data representations are handled in FTP by a user specifying a
representation type. The type may also imply a transfer byte size.
For example, in ASCII representation, the transfer byte size should
be 8 bits, and any other byte size specification will result in

cancellation of the transfer request. In image and Local Byte
representations any byte size is possible. The following data
representation types are currently defined in FTP:

1. ASCII   The sender converts data from its internal character
                representation to the standard NVT ASCII form. The
                receiver converts the data from the standard form to
                its own internal form. The data is transferred in
                the standard form. The transfer byte size must be 8
                bits. This type would be used for transfer of text
                files. This is the default type, and it is recom-
                mended that this type be implemented by all.

2. EBCDIC   The sender transfers data using the EBCDIC character
                code and 8-bit transfer byte size. This type may be
                used for efficient transfer of EBCDIC files between
                systems which use EBCDIC for their internal character
                representation.

3. Image   The sender transforms data from contiguous bits to
                bytes for transfer. The receiver transforms the
                bytes into bits, storing them contiguously indepen-
                dent of the byte size chosen for data transfer. With
                record structure and block mode, the server might
                need to pad each record for convenient storage. This
                padding is allowed at the end of a record, and should
                be remembered by the server so it will be stripped
                off when the file is retrieved by the user. The pad-
                ding transformation should be well publicized by the
                server in case the user processes his file at the
                server site. Typical uses for the Image type are
                transfer of executable programs between like
                machines, and transfer of binary (non-text) data. It
                is recommended that this type be implemented by all
                for some byte size, preferably including the 8 bit
                byte size.

4. Local Byte   This representation allows for efficient storage,
                use, and retrieval of data. The manner in which data
                is to be transformed depends on the byte size for
                data transfer, and the particular HOST being used.
                The transformation scheme for different byte size is
                to be well publicized by all server sites. This
                transformation shall be invertible (i.e., if a file
                is stored using a certain transfer byte size, an
                identical file must be retrievable using the same
                byte size and representation type). It is the user's
                responsibility to keep track of the representation

type and byte size used for his transfer. Typical
uses of the Local Byte type are in efficient storage
and retrieval of files, and transfer of structured
binary data. This type may be identical to the Image
type for byte size which are integral multiples of or
factors of the computer word length.

Representation type may also be affected by another attribute, the
format. For example, some printers can use ASA (Fortran) vertical
format control procedures to transform printed data of type ASCII or
EBCDIC. Currently format may take one of two values.

1. Unformatted The representation type as specified is unaffected by
                any format transformations. This is the default
                value.

2. Printfile   The server is to transform data of either ASCII or
                EBCDIC type in accordance with ASA (Fortran) vertical
                format control standards. The data is to be
                transferred in 8-bit bytes.

A discussion of the ASA vertical format control appears in NWG/RFC
189, Appendix C, and in Communications of the ACM, Vol. 7, No. 10, p.
606, October 1964. According to the ASA vertical format control
standards, the first character of a formatted record is not printed
but determines vertical spacing as follow:

    Character   Vertical Spacing before printing

     Blank   One line
       0   Two lines
       1   To first line of the next page
       +   No advance

In addition to the above four, there are more characters (defined in
Appendix C, RFC 189) which represent an IBM extension to the ASA
standard.

It should be noted that a serving host need not accept all represen-
tation types and/or byte sizes, but it must inform the user request-
ing an unacceptable type or size of this fact by sending an appropri-
ate reply.

III.C. File Structure and Transfer Modes

The only file structures supported directly in FTP at the present
time are record structures. However, the use of record structures is
not mandatory. A user with no record structure in his file should be

able to store and retrieve his file at any HOST. A user wishing to
transmit a record structured file must send the appropriate FTP
'STRU' command (the default assumption is no record structure). A
serving HOST need not accept record structures, but it must inform
the user of this fact by sending an appropriate reply. Any record
structure information in the data stream may subsequently be dis-
carded by the receiver.

All data transfers must end with an EOF. The EOF is defined by the
data transfer mode. For files that have record structures, an EOR is
also defined by the transfer mode. Only the transfer modes and
representation type combinations that have EOR defined may be used
for transfer of files with record structures. Records may be of zero
length but they must be contained in file boundaries. The relation-
ship between files and records is hierarchical but an EOF does not
imply an EOR.

The following data transfer modes are defined in FTP:

1. Stream   The file is transmitted as a stream of bytes of the
                 specified byte size. The EOF is signaled by closing
                 the data connection. Any representation type and
                 byte size may be used in the stream mode with file
                 structure, but use of record structure limits the
                 type to ASCII or EBCDIC with or without Printfile
                 format. The convention is that the ASCII character
                 CR (Carriage Return, Code 15 (octal)) followed by LF
                 (Line Feed, Code 12 (octal)) indicates an EOR for
                 ASCII representation type, and the EBCDIC character
                 NL (New Line, Code 15 (hex)) indicates an EOR for
                 EBCDIC type. This is the default mode, and it is
                 recommended that this mode be implemented by all.

2. Text   The file is ASCII text transmitted as a sequence of
                 8-bit bytes in the ASCII representation type, and
                 optional Printfile format. Record structures are
                 allowed in this mode. The EOR and EOF are defined by
                 the presence of special "TELNET-control" codes (,ost
                 significant bit set to one) in the data stream. The
                 EOR code is 192 (octal 300, hex CO). The EOF code is
                 193 (octal 301, hex C1). The byte size for transfer
                 is 8 bits.

(For ASCII type, text and stream modes are almost identical.)

Comparing the two, the advantages of "stream" mode are:

   1) The receiver need not scan the incoming bytes.

   2) It is usable with all data types.

and the disadvantages are:
   1) Closing the data connection under error conditions can be
      misconstrued as an EOF in stream mode when in fact the data
      transfer was interrupted. In text mode the EOF is sent expli-
      citly.

   2) If record structure is specified in stream mode then CRLF
      implies EOR, and in order for CRLF to be sent as valid data it
      must be transformed, e.g., into CR NUL LF or LF CR.

3. Block   The file is transmitted as a series of data blocks
                  preceded by one or more header bytes. The header
                  bytes contain a count field, and descriptor code.
                  The count field indicates the total length of the
                  data block in bytes, thus marking the beginning of
                  the next data block (there are no filler bits). The
                  descriptor code defines last file block (EOF), last
                  record block (EOR), restart marker (see Section
                  III.D), or suspect data (i.e., the data being
                  transferred is suspected of errors and is not reli-
                  able). Record structures are allowed in this mode,
                  and any representation type or byte size may be used.

                  The header consists of the smallest integral number
                  of bytes whose length is greater than or equal to 24
                  bits. Only the _least_ significant 24 bits (right-
                  justified) of header shall have information; the
                  remaining most significant bits are "don't care"
                  bits. Of the 24 bits of header information, the 16
                  low order bits shall represent byte count, and the 8
                  high order bits shall represent descriptor codes as
   shown below.

                        Integral data bytes >= 24
                  +--------------+--------------+-------------+
                  | Don't care | Descriptor | Byte Count |
                  | 0 to 231 bits| 8 bits | 16 bits |
                  +--------------+--------------+-------------+

The following descriptor codes are assigned:

Code   Meaning
----   -------
 0   An ordinary block of data.
 1   End of data block is EOR.
 2   End of data block is EOF.
 3   Suspected errors in data block.
 4   Data block is a restart marker.

In the use of block mode it is possible for two or
more conditions requiring different descriptor codes
(suspected errors and either end of record or end of
file) to exist simultaneously. Such a possibility
may be handled by sending a separate EOR or EOF block
with a zero byte count. (This is allowed by the pro-
tocol.)

The restart marker is embedded in the data stream as
an integral number of 8-bit bytes (representing
printable ASCII characters) right-justified in an
integral number of data bytes greater than 8 bits.
For example if the byte size is 7 bits, the restart
marker byte would be one byte right-justified per two
7-bit bytes as shown below:

     Two 7-bit bytes
+----------+------------+
| | Marker Char|
| | 8 bits |
+----------+------------+
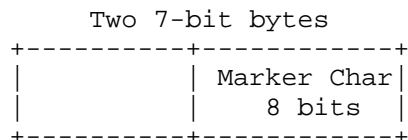
For byte size of 16 bits or more, two or more marker
bytes shall be packed right-justified. The end of
the marker may be delimited by the character SP (code
32.). If marker characters do not exactly fit an
integral byte, the unused character slots should con-
tain the ASCII character SP (code 32.). For example,
to transmit a six character marker in a 36-bit byte
size, the following three 36-bit bytes would be sent:

```
            +------------+------------+--------------+
            | Don't care | Descriptor |              |
            |   12 bits  |  code=4    | Byte count=2 |
            +------------+------------+--------------+


            +----+--------+--------+--------+--------+
            |    | Marker | Marker | Marker | Marker |
            |    | 8 bits | 8 bits | 8 bits | 8 bits |
            +----+--------+--------+--------+--------+


            +----+--------+--------+--------+--------+
            |    | Marker | Marker | SP     | SP     |
            |    | 8 bits | 8 bits | 8 bits | 8 bits |
            +----+--------+--------+--------+--------+
```

4. Hasp

> The file is transmitted as a sequence of 8-bit bytes
> in the standard Hasp-compressed data format (document
> to be issued by Bob Braden, UCLA). This mode
> achieves considerable compression of data for print
> files. Record structures are allowed in the Hasp
> mode.

The following matrix summarizes the legal combinations of file
transfer parameters. The decimal integers represent legal byte sizes
for each particular STRU-MODE-TYPE-FORM grouping absence of a number
implies illegality. Note that HASP mode is not included since it has
never been defined.

```
         STRU                F                |          R
         +-----------------------------+-----+-----+------+
TYPE     |\ MODE                       |     |     |      |
         | \                           |     |     |      |
         |  \      S       T       B   |  S  |  T  |  B   |
         | FORM +--------+-----+--------+-----+-----+------+
    A    |  U  |   8    |  8  |   8    |  8  |  8  |  8   |
         |     +--------+-----+--------+-----+-----+------+
         |  P  |   8    |  8  |   8    |  8  |  8  |  8   |
  ----+------+--------+-----+--------+-----+-----+------+
    E    |  U  |   8    |     |   8    |  8  |     |  8   |
         |     +--------+-----+--------+-----+-----+------+
         |  P  |   8    |     |   8    |  8  |     |  8   |
  ----+------+--------+-----+--------+-----+-----+------+
    I    |  U  | 1-255  |     | 1-255  |     |     |1-255 |
  ----+------+--------+-----+--------+-----+-----+------+
    L    |  U  | 1-255  |     | 1-255  |     |     |1-255 |
  ----+------+--------+-----+--------+-----+-----+------+
```

III.D Error Recovery and Restart

    There is no provision for detecting bits lost or scrambled in data
    transfer. This issue is perhaps handled best at the NCP level where
    it benefits most users. However, a restart procedure is provided to
    protect user from system failures (such as failure of either HOST,
    FTP-process, or the IMP subnet).

    The restart procedure is defined only for the block mode of data
    transfer. It requires the sender of data to insert a special marker
    code in the data stream with some marker information. The marker
    information has meaning only to the sender, but must consist of
    printable ASCII characters. The printable ASCII characters are
    defined to be octal codes 41 through 176 (i.e., not including codes 0
    through 37 and the characters SP and DEL). The marker could
    represent a bit-count, a record-count, or any other information by
    which a system may identify a data checkpoint. The receiver of data,
    if it implements the restart procedure, would then mark the
    corresponding position of this marker in the receiving system, and
    return this information to the user.

    In the event of a system failure, the user can restart the data
    transfer by identifying the marker point with the FTP restart pro-
    cedure. The following examples illustrate the use of the restart
    procedure.

1.   When server is the sender of data, the server-FTP process inserts
        an appropriate marker block in the data stream at a convenient
        data point. The user-FTP process, receiving the data, marks the
        corresponding data point in its file system and conveys the last
        known sender and receiver marker information to the user. In the
        event of system failure, the user or user-FTP process restarts
        the server at the last server marker by sending a restart command
        with the server's marker code as its argument. The restart com-
        mand is transmitted over the TELNET connection and is immediately
        followed by the command (such as store or retrieve) which was
        being executed when the system failure occurred.

2.   When user is the sender of data, the user-FTP process inserts the
        appropriate marker block in the data stream. The server-FTP pro-
        cess, receiving the data, marks the corresponding data point in
        its file system. The server does not store this marker but con-
        veys the last known sender and receiver marker information to the
        user over the TELNET connections by appropriate reply codes. The
        user or the user-FTP process then restarts transfer in a manner
        identical to that described in the first example.

IV. FILE TRANSFER FUNCTIONS

The TELNET connections on which FTP commands and replies are
transmitted are initiated by the user-FTP process via an ICP to a
standard server socket. FTP commands are then transmitted from user
to server, and replies are transmitted from server to user. The user
file transfer functions involve sending the FTP commands, interpret-
ing the replies received and transferring data over the data connec-
tion in the specified manner. The server file transfer functions
involve accepting and interpreting FTP commands, sending replies,
setting up the data connection, and transferring data.

IV.A   FTP Commands

   FTP commands are ASCII strings terminated by the ASCII character
   sequence CRLF (Carriage Return followed by Line Feed). The command
   codes themselves are ASCII alphabetic characters terminated by the
   ASCII character 'space' (octal code 40). For convenience, the com-
   mand codes are defined to be four (or less) ASCII alphanumeric char-
   acters (including both upper and lower case alphabetic characters).
   The command codes and the semantics of commands are described in this
   section, but the detailed syntax of commands is specified in Section
   V.B, the reply sequences are discussed in Section V.C, and scenarios
   illustrating the use of commands are provided in Section V.D.

   FTP commands may be partitioned as those specifying access-control
   identifiers, data transfer parameters, or FTP service requests. Cer-
   tain commands (such as ABOR, STAT, BYE) may be sent over the TELNET
   connections while a data transfer is in progress. Some servers may
   not be able to monitor the TELNET and data connections simultane-
   ously, in which case these commands should be preceded by a TELNET
   SYNC to awaken the server. (For other servers this may not be neces-
   sary and the SYNC will be ignored.)

IV.A.1   Access Control Commands

   The following commands specify access control identifiers (command
   codes are shown in parentheses).

      User name (USER) - The argument field is an ASCII string identify-
      ing the user. The user identification is that which is required
      by the server for access to its file system. This command will
      normally be the first command transmitted by the user after the
      TELNET connections are made (some servers may require this).
      Additional identification information in the form of a password
      and/or an account command may also be required by some servers.
      Servers may allow a new USER command to be entered at any point in
      order to change the accounting information. All parameters are
      unchanged and any file transfer in progress is completed under the
      old account.

      Password (PASS) - The argument field is an ASCII string identify-
      ing the user's password. This command must be immediatly preceded
      by the user name command, and, for some sites, completes the user'
      s identification for access control. Since password information
      is quite sensitive, it is desirable in general to "mask" it or
      suppress type out. It appears that the server has no foolproof
      way to achieve this. It is therefore the responsibility of the
      user-FTP process to hide the sensitive password information.

Account (ACCT) – The argument field is an ASCII string identifying
the user's account. The command is not necessarily related to the
USER command, as some sites may require an account for login and
others only for specific access, such as storing files. In the
latter case the command may arrive at any time. There are two
reply codes to differentiate these cases for the automaton: When
account information is required for login and the server receives
another command which he buffers, the legal response is reply code
331 when an account is required for a specific transfer requested,
the reply code 433 is returned and the request command is flushed.

Reinitialize (REIN) – This command terminates a USER, flushing all
I/O and account information, except to allow any transfer in pro-
gress to be completed. All parameters are reset to the default
setting and the TELNET connection is left open. A USER command is
expected to follow.

Logout (BYE) – This command terminates a USER and if file transfer
is not in progress, closes the TELNET connection. If file
transfer is in progress, the connection will remain open for
result response and will then close. For "hot card-reader" mode
the REIN command should be used instead.

An unexpected close on the TELNET connection will cause the server
to take the effective action of an abort (ABOR) and a logout
(BYE).

IV.A.2 Transfer Parameter Commands

All data transfer parameters have default values, and the commands
specifying data transfer parameters are required only if the default
parameter values are to be changed. The default value is the last
specified value, or if no value has been specified, the standard
default value as stated here. This implies that the server must
"remember" the applicable default values. The commands may be in any
order except that they must precede the FTP service request. The
following commands specify data transfer parameters

Byte size (BYTE) – The argument is an ASCII-represented decimal
integer (1 through 255), specifying the byte size for the data
connection. The default byte size is 8 bits. The byte size is
always 8 bits in the ASCII and EBCDIC representation types. A
server may reject specific byte size/type combinations by sending
an error reply code in response to a transfer request command.

Data socket (SOCK) – The argument is a HOST-socket specification
for the data socket to be used in data connection. There may be
two data sockets, one from server to user and the other for user

to server data transfer. An odd socket number defines a send
socket and an even socket number defines a receive socket. The
default HOST is the user HOST to which TELNET connections are
made. The default data sockets are (U+4) and (U+5) where U is the
socket number used in the TELNET ICP and the TELNET connections
are on sockets (U+2) and (U+3).

Listen (LSTN) - The argument is a single ASCII character code to
specify the direction of the socket that the server must allocate
for use as a data connection. The server is to "listen" on the
allocated socket when an appropriate transfer command is given.
The following codes are assigned:

        S - send
        R - receive

Representation Type (TYPE) - The argument is a single ASCII char-
acter code specifying the representation types described in Sec-
tion III.B. The following codes are assigned for type:

        A - ASCII
        I - Image
        L - Local Byte
        E - EBCDIC

The default representation type is ASCII.

Format (FORM) - The argument is a single ASCII character code
specifying the formats described in Section III.B. The following
codes are assigned for format:

        U - Unformatted
        P - Printfile

The default format is Unformatted.

File Structure (STRU) - The argument is a single ASCII character
code specifying file structure described in Section III.C. The
following codes are assigned for structure:

        F - File (no ecord structure)
        R - Record structure

The default structure is File (ie. no records).

Transfer Mode (MODE) - The argument is a single ASCII character
code specifying the data transfer modes described in Section
III.C. The following codes are assigned for transfer modes:

```
              S - Stream (bytes, close is EOF)
              B - Block (header with descriptor and count)
              T - Text (TELNET control code for EOR, EOF)
              H - Hasp (specially formatted compressed data)
```

     The default transfer mode is Stream.

IV.A.3 FTP Service Commands

   The FTP service commands define the file transfer or the file system
   function requested by the user. The argument of an FTP service com-
   mand will normally be a pathname. The syntax of pathnames must con-
   form to server site conventions (with standard defaults applicable),
   except that ASCII characters must be used (in conformance with the
   TELNET Protocol). The suggested default handling is to use the last
   specified device, directory or file name, or the standard default
   defined for local users. The command may be in any order except that
   a "rename from" command, must be followed by a "rename to" command,
   and some servers may require an "allocate" command before a "store"
   command. The data, when transferred in response to FTP service
   commands, shall always be sent over the data connection. The follow-
   ing commands specify FTP service requests:

     Retrieve (RETR) - This command achieves the transfer of a copy of
     the file specified in the pathname, from server to user site. The
     status and contents of the file at the server site shall be unaf-
     fected.

     Store (STOR) - This command achieves the transfer of a copy of a
     file from user to server site. If the file specified in the path-
     name exists at the server site, then its contents shall be
     replaced by the contents of the file being transferred. A new
     file is created at the server site if the file specified in the
     pathname does not already exist.

     Append (with create) (APPE) - This command achieves the transfer
     of data from using to serving site. If the file specified in the
     pathname exists at the server site, then the data transferred
     shall be appended to that file, otherwise the file specified in
     the pathname shall be created at the server site.

     Allocate (ALLO) - This command may required by some servers to
     reserve sufficient storage to accommodate the new file to be
     transferred. The argument field shall be a decimal integer
     representing the number of bytes (of size specified by the byte
     size command) of storage to be reserved for the file. This

command shall be followed by a store or append command. The ALLO
command should be treated as a NO-OP (no operation) by those
servers which do not require that the maximum size of the file be
declared beforehand.

Restart (REST) - The argument field represents the server marker
at which file transfer is to be restarted. This command does not
cause file transfer but "spaces" over the file to the specified
data checkpoint. This command shall be immediately followed by
the appropriate FTP service command which shall cause file
transfer to resume.

Rename from - (RNFR) - This command specifies the file which is to
be renamed. This command must be immediately followed by a
"rename to" command specifying the new file pathname.

Rename to (RNTO) - This command specifies the new pathname of the
file specified in the immediately preceding "rename from" command.
Together the two commands cause a file to be renamed.

Abort (ABOR) - This command indicates to the server to abort the
previous FTP service command and any associated transfer of data.
The abort command should be preceded by the TELNET SYNCH condition
(indicated by the combination of the DATA MARK and the INS). No
action is to be taken if the previous command has been completed
(including data transfer). The TELNET connections are not to be
closed by the server, but the data connection may be closed. An
appropriate reply should be sent by the server.

Delete (DELE) - This command causes the file specified in the
pathname to be deleted at the server site. If an extra level of
protection is desired (such as the query, "Do you really wish to
delete?"), it should be provided by the user-FTP process.

List (LIST) - This command causes a list to be sent from server to
user site. If the pathname specifies a directory, the server
should transfer a list of files in the specified directory. If
the pathname specifies a file then server should send current
information on the file. A null argument implies the user's
current working or default directory. The data transfer is over
the data connection in type ASCII or type EBCDIC. (It is the user
's responsibility to ensure the correct parameters.)

NList (NLST) - This command causes a directory listing to be sent
from server to user site. The pathname should specify a directory
and the server will return a stream of names of files and no other
information. The data will be transferred in ASCII or EBCDIC type
over the data connection as valid pathname strings separated by

CRLF. This command will allow automatic copying of an entire
directory when used with the appropriate transfer commands.

Status (STAT) - This command shall cause a status response to be
sent over the TELNET connection in form of a reply. The command
may be sent during a file transfer (preceded by a TELNET SYNC) in
which case the server will respond with the status of the opera-
tion in progress, or it may be sent between file transfers. In
the latter case the command may have an argument field such as a
pathname. If the argument is a pathname, the command is analogous
to the "list" command except that data shall be transferred in
ASCII on the TELNET connection. If a partial pathname is given,
the server may respond with a list of file names or attributes
associated with that specification. If no argument is given, the
server should return general status information about the server
FTP process. This should include current values of all transfer
parameters and the status of connections.

Help (HELP) - This command shall cause the server to send helpful
information regarding its implementation status over the TELNET
connection to the user. The command may take an argument (e.g.
any command name) and return more specific information as a
response. The reply is type 100, general system status. It is
suggested that HELP be allowed before entering a USER command.

Mail File (MLFL) - The intent of this command is to enable a user
site to mail data (in form of a file) to another user at the
server site. It should be noted that the files to be mailed are
transmitted via the data connection in ASCII or EBCDIC type. (It
is the user's responsibility to ensure that the type is correct.)
These files should be appended to the destination user's mail by
the server in accordance with serving HOST mail conventions. The
mail may be marked as sent from the particular using HOST and the
user specified by the 'USER' command. The argument field may con-
tain one or more system or NIC idents (it is recommended that mul-
tiple ident be allowed so the same mail can easily be sent to
several users), or it may be empty. If the argument field is
empty or blank (one or more spaces), then the mail is destined for
a printer or other designated place for site mail. A NIC ident
refers to the standard identification described in the NIC Direc-
tory of Network Participants. A serving host may keep a table
mapping NIC indents into system idents, although NIC idents are
not required in the implementation. A system ident is the user's
normal identification at the serving host. The use of system
idents would allow a network user to send mail to other users who
do not have NIC identification but whose system ident is known.

Mail (MAIL) - This command allows a user to send mail that is not
in a file over the TELNET connection. The argument field may con-
tain one or more system or NIC idents, or it may be empty. The
idents are defined as above for the MLFL command. After the
'MAIL' command is received, the server is to treat the following
lines as text of the mail sent by the user. The mail text is to
be terminated by a line containing only a single period, that is,
the character sequence ".CRLF" in a new line. It is suggested
that a modest volume of mail service should be free; i.e., it may
be entered before a USER command.

IV.A.4 Miscellaneous Commands

NoOP (NOOP) - This command does not affect any parameters or pre-
viously entered command. The server simply sends a no-op reply.

Quote (QUOT) - This command allows the user to talk directly to
the FTP-server. After parsing this command, the user-FTP process
will pass without examination all succeeding liners until the NQUO
command is received. Between these two commands the server will
respond appropriately to his implementation and the user's
requests.

NoQuote (NQUO) - This command returns the user and server
processes to normal interactive mode. Both QUOT and NQUO have
reply codes to be sent by th server process to the user process to
ensure agreement on the current mode.

The quote commands provide a convenient method of testing server-
implemented experimental commands. The names of the latter should
begin with an X, and can be listed in the system HELP reply. It
should be noted that the official command set is expandable; sugges-
tions should go first to Alexander A. McKenzie (BBN).

IV.B FTP Replies

The server sends FTP replies over the TELNET connection in response
to user FTP commands. The FTP replies constitute the acknowledgment
or completion code (including errors). The FTP-server replies are
formatted for human or program interpretation. Single line replies
consist of a leading three-digit numeric code followed by a space,
followed by a one-line text explanation of the code. For replies
that contain several lines of text, the first line will have a lead-
ing three-digit numeric code followed immediately by the ASCII char-
acter "-" (Hyphen, Code 55 (octal)) and possibly some text. All
succeeding continuation lines except the last are constrained not to
begin with three digits; the last line must repeat the numeric code
of the first line and be followed immediately by a space.

For example:

          100-First Line
          Continuation Line
          Another Line
          100 Last Line

The numeric codes are assigned by groups and for ease of interpreta-
tion by programs in a manner consistent with other protocols such as
the RJE protocol. The three digits of the code are to be interpreted
as follows:

a) The first digit specifies type of response as indicated below:

     000 These replies are purely informative and constitute neither a
         positive nor a negative acknowledgment.

     1xx Informative replies to status inquiries. These constitute a
         positive acknowledgment to the status command.

     2xx Positive acknowledgment of previous command or other success-
         ful action.

     3xx Incomplete information. Activity cannot proceed without
         further specification and input.

     4xx Unsuccessful reply. The request is correctly specified but
         the server is unsuccessful in correctly fulfilling it.

     5xx Incorrect or illegal command. The command or its parameters
         were invalid or incomplete from a syntactic viewpoint, or the
         command is inconsistent with a previous command. The command
         in question has been completely ignored.

     6xx-9xx Reserved for future expansion.

b) The second digit specifies the general category to which the
   response refers:

   x00-x29 General purpose replies, not assignable to other
   categories.

   x30 Primary access. Informative replies to the "log-on" attempt.

   x40 Secondary access. The primary server is commenting on its
   ability to access a secondary service.

   x5x FTP results

   x6x RJE results.

   x7x-x9x Reserved for future expansion.

c) The final digit specifies a particular message type. Since the
   code is designed for an automation process to interpret, it is
   not necessary for every variation of a reply to have a unique
   number. Only the basic meaning of replies need have unique
   numbers. The text of a reply can explain the specific reason for
   that reply to a human user.

   Each TELNET line delimited by a numeric code and CRLF (or group
   of text lines bounded by coded lines) that is sent by the server
   is intended to be a complete reply message. It should be noted
   that the text of replies is intended for a human user. Only the
   reply codes and in some instances the first line of text are
   intended for programs.

The assigned reply codes relating to FTP are:

000 General information message (site, time of day, etc.).
010 Message from system operator.
030 Server availability information.
050 FTP commentary or user information.
100 System status reply.
110 System busy doing...
150 File status reply
151 Directory listing reply.
200 Last command received correctly.
201 An ABORT has terminated activity, as requested.
202 Abort request ignored, no activity in progress.
230 User is "logged in". May proceed.
231 User is "logged out". Service terminated.
232 Logout command noted, will complete when transfer done.
233 User is "logged out". Parameters reinitialized.

250 FTP file transfer started correctly.
251 FTP Restart-marker reply.

        Text is : MARK yyyy = mmmm
        where yyyy is user's data stream marker (yours)
        and mmmm is server's equivalent marker (mine)
        (Note the spaces between the markers and '=')

252 FTP transfer completed correctly.
253 Rename completed.
254 Delete completed.
255 FTP server data socket reply

        Text is: SOCK nnnn
        where nnnn is a decimal integer representing
        the server socket for data connection

256 Mail completed.
300 Connection greeting message, awaiting input.
301 Current command incompleted (no CRLF for long time).
330 Enter password
331 Enter account (if account required as part of login
    sequence).
350 Enter mail, terminate by a line with only a '.'
400 This service not implemented.
401 This service not accepting user now, goodbye.
430 Log-on time or tries exceeded, goodbye.
431 Log-on unsuccessful. Usre and/or password invalid.
432 User not valid for this service.
433 Cannot transfer files without valid account. Enter account.
434 Log-out forced by operator action. Phone site.
435 Log-out forced by system problem.
436 Service shutting down, goodbye.
450 FTP: File not found.
451 FTP: File access denied to you.
452 FTP: File transfer incomplete, data connection closed.
453 FTP: File transfer incomplete, insufficient storage space.
454 FTP: Cannot connect to your data socket.
455 FTP: File system error not covered by other reply codes.
456 FTP: Name duplication rename failed.
457 FTP: Transfer parameters in error.
500 Last command line completely unrecognized.
501 Syntax of last command is incorrect.
502 Last command incomplete, parameters missing.
1234567890123456789012345678901234567890123456789012345678901234567890123456789012
503 Last command invalid (ignored), illegal parameter combination.
504 Last command invalid, action not possible at this time.
505 Last command conflicts illegally with previous command(s).

506 Requested action not implemented by the server.
507 Catchall error reply.
550 Bad pathname specification (e.g., syntax error).

V. DECLARATIVE SPECIFICATIONS

   In order to make FTP workable without needless error messages, the
   following minimum implementation is required for servers:

TYPE -- ASCII (with 8-bit bytes)
                MODE -- Stream
                STRUCTURE -- File
                             Record (with ASCII type and CRLF for EOR)
                FORM -- Unformatted
                COMMANDS -- USER, BYE, SOCK
                            TYPE, BYTE, MODE, STRU, FORM
                               for the default values
                            RETR, STOR
                            NOOP

   The initial default values for transfer parameters are:

      TYPE -- ASCII
                    BYTE -- 8
                    MODE -- Stream
                    STRU -- File
                    FORM -- Unformatted

V.A Connections

   The server-FTP process at the server site shall "listen" on Socket 3,
   via its server-TELNET. The user or user-FTP process at the user site
   shall initiate the full-duplex TELNET connections via its user-TELNET
   performing the ARPANET standard initial connection protocol (ICP) to
   server socket 3. Servers may specify that interaction over the TEL-
   NET connections be line-at-a-time with local echo. The server is not
   obliged to provide remote echo and may ignore TELNET control charac-
   ters; he should not, however, return error response to the latter.
   All editing of command lines similarly must be local. The TELNET
   connections shall be closed by the user site upon completion of use
   and receipt of the last server reply.

   The user site must "listen" on the specified data socket or sockets
   (a send and/or a receive socket). The server site shall initiate the
   data connection using the specified data socket and byte size. The
   direction of data connection and the data socket used shall be

determined by the FTP service command. The server shall send a reply
to the user indicating the server data socket so that the user may
ensue the security of data transfer. This can be done at any time
prior to the first transfer of data over a data connection. It
should be emphasized that the user-FTP should not wait for a 255
(server data socket) reply before doing the "listen", since there is
no guarantee that the reply will arrive before the user site receives
the initiating RFC. The security check can be done when the reply
arrives and the data connection closed if it was made to a socket
other than the one specified.

The data connection shall be closed by the server site under the con-
ditions described in Section III.A. If the server wishes to close
the connection in modes where that is not required, it is recommended
that the close be sent immediately after the file transfer is com-
pleted rather than after a new transfer command is received, because
the user or server may have to test the state of the socket before
doing a "listen" or "init". The server should in general send a
reply before closing the data connection to avoid problems at the
user end, though, for reasons stated above, the user-FTP should not
wait for the reply before doing his close.

V.B Commands

The commands are ASCII character strings transmitted over the TELNET
connections as described in section IV.A. The command functions and
semantics are described in sections IV.A.1, IV.A.2, IV.A.3, and
IV.A.4. The command syntax is specified here.

The commands begin with a command code followed by an argument field.
The command codes are four or less ASCII alphabetic characters.
Upper and lower case alphabetic characters are to be treated identi-
cally. Thus any of the following may represent the retrieve command:

   RETR   Retr   retr   ReTr   rETr

This also applies to any symbols representing parameters values, such
as A or a for ASCII TYPE. The command codes and the argument fields
are separated by one or more spaces.

The argument field consists of a variable length ASCII character
string ending with the character sequence CRLF (Carriage Return
immediately followed by Line Feed). In the following section on syn-
tax it should be stressed that all characters in the argument field
are ASCII characters. Thus a decimal integer shall mean an ASCII
represented decimal integer.

The following are all the currently defined FTP commands:

    USER <username> CRLF

    PASS <password> CRLF

    ACCT <acctno> CRLF

    REIN CRLF

    BYE CRLF

    BYTE <byte size> CRLF

    SOCK <HOST-socket> CRLF

    LSTN <direction> CRLF

    TYPE <type code> CRLF

    FORM <form code> CRLF

    STRU <structure code> CRLF

    MODE <mode code> CRLF

    RETR <pathname> CRLF

    STOR <pathname> CRLF

    APPE <pathname> CRLF

    ALLO <decimal integer> CRLF

    REST <marker> CRLF

    RNFR <pathname> CRLF

    RNTO <pathname> CRLF

    ABOR CRLF

    DELE <pathname> CRLF

    LIST <pathname> CRLF

    NLST <pathname> CRLF

```
STAT <pathname> CRLF

HELP <string> CRLF

MLFL <users> CRLF

MAIL <users> CRLF

NOOP CRLF

QUOT CRLF

NQUO CRLF
```

The syntax of the above argument fields (using BNF notation where applicable) is:

```
<username> ::= <string>

<password> ::= <string>

<acctno> ::= <string>

<string> ::= <empty>/<char>/<char><string>

<char> ::= any of the 128 ASCII characters except CR and LF.

<marker> ::= <pr string>

<pr string> ::= <empty>/<pr char>/<pr char> <pr string>

<pr char> ::= any ASCII code 33 through 126

<byte size> ::= any decimal integer 1 through 255.

<HOST-socket> ::= <socket>/HOST number>,<socket>

<HOST number> ::= a decimal integer specifying an ARPANET HOST

<socket> ::= decimal integer between 0 and (2**32)-1

<direction> ::= S/R

<form code> ::= U/P

<type code> ::= A/E/I/L

<structure code> ::= F/R
```

```
<mode code> ::= S/B/T/H

<pathname> ::= <string>

<decimal integer> ::= <digit>/<digit><decimal integer>

<digit> ::= 0|1|2|3|4|5|6|7|8|9

<empty> ::= the null string (specifies use the default).

<users> ::= <user>|<user,<users>

<user> ::= <empty>|<NIC ident>|<sys ident>

<NIC ident> ::= <string>

<sys ident> ::= <string>
```

V.C Sequencing of Commands and Replies

   The communication between the user and server is intended to be an
   alternating dialogue. As such, the user issues an FTP command and
   the server responds with a prompt primary reply. The user should
   wait for this initial primary success or failure response before
   sending further commands.

   A second type of reply is sent asynchronously with respect to user
   commands. These replies may, for example, report on the progress or
   completion of file transfer and as such are secondary replies to file
   transfer commands.

   The third class of replies are informational and spontaneous replies
   which may arrive at any time. These replies are listed below as
   spontaneous.

COMMAND-REPLY CORRESPONDENCE TABLE

```
COMMAND               SUCCESS        FAIL
-------               -------        ----
USER                  230,330        430-432,500-505,507
PASS                  230,331        430-432,500-507
ACCT                  230            430-432,500-507
REIN                  232,233        401,436,500-507
    Secondary Reply   300
BYE                   231,232        430-432,500-505,507
BYTE                  200,331        500-507
SOCK                  200,331        500-505,507
LSTN                  255,331        500-507
TYPE                  200,331        500-507
FORM                  200,331        500-507
STRU                  200,331        500-507
MODE                  200,331        500-507

RETR                  250,331        433,450,451,454,455,500-505,507,550
    Secondary Reply   252            452
STOR                  250,331        433,451,454,455,457,500-505,507,550
    Secondary Reply   252            452,453
APPE                  250,331        433,451,454,455,457,500-507,550
    Secondary Reply   252            452,453
ALLO                  200,331        500-507
REST                  200,331        500-507
RNFR                  200,331        433,450,451,455,500-507,550
RNTO                  253,331        433,450,451,455,456,500-505,507,550
ABOR                  201,202,331    500-507
DELE                  254,331        433,450,451,455,500-507,550
LIST                  250,331        433,450,451,454,455,457,500-507,550
    Secondary Reply   252            452
NLST                  250,331        433,450,451,454,455,457,500-507
    Secondary Reply   252            452
STAT                  100,110,150,   450,451,454,455,500-507,550
                      151,331
HELP                  000,030,050,   500-507
                      331
MLFL                  250,331        433,450,451,454,455,457,500-507
    Secondary Reply   252            452,453
MAIL                  331,350        433,450,451,455,500-507
    Secondary Reply   256
NOOP                  200            500-505,507
QUOT                  200,331        500-507
NQUO                  200            500-505,507

Spontaneous           0xx,300,301    400,401,434-436
Replies               251,255
```

V.D Typical FTP Scenarios

   1. TIP User wanting to transfer file from HOST X to local printer:

      a) TIP user opens TELNET connections by ICP to HOST X, socket 3.

      b) The following commands and replies are exchanged:

         TIP   HOST X
         ---   ------

         USER username CRLF ---------->
         <----------330 Enter Password CRLF

         PASS password CRLF ---------->
         <----------230 User logged in CRLF

         SOCK 65538 CRLF   ---------->
         <----------200 Command received OK CRLF

         RETR this.file CRLF ---------->
         <----------255 SOCK 5533 CRLF

         (HOST X initiates data connection to
          TIP socket 65538, i.e., PORT 1 receive)

         <----------250 File transfer started

         BYE CRLF   ----------------->
         <----------252 File transfer completed

      c) HOST X closes the TELNET and data connections.

      Note: The TIP user should be in line mode.

   2. User at HOST U wanting to transfer files to/from HOST S:

      In general the user would communicate to the server via a mediat-
      ing user-FTP process. The following may be a typical scenario.
      The user-FTP prompts are shown in parentheses, '---->' represents
      commands from HOST U to HOST S, and '<----' represents replies
      from HOST S to HOST U.

```
Local Commands by User              Action Involved
---------------------              --------------

ftp (host) multics CR               ICP to HOST S, socket 3,
                                    establishing TELNET connections.
username Doe CR                     USER Doe CRLF ---->
                                    <---- 330 password CRLF
password mumble CR                  PASS mumble CRLF ---->
                                    <---- 230 Doe logged in. CRLF
retrieve (local type) ASCII CR
(local pathname) test 1 CR          User-FTP opens local file in ASCII.
(for.pathname) test.p11 CR          RETR test.p11 CRLF
                                    <---- 255 SOCK 1233 CRLF
                                    Server makes data connection to (U+4).
                                    <---- 250 File transfer starts CRLF
                                    <---- 252 File transfer complete CRLF
type ImageCR                        TYPE I CRLF ---->
                                    <---- 200 Command OK CRLF
byte 36CR                           BYTE 36 CRLF ---->
                                    <---- 200 Command OK CRLF
store (local type) image CR
(local pathname) file dump CR       User-FTP opens local file in Image.
(for.pathname) >udd>cn>fd CR        STOR >udd>cn>fd CRLF ---->
                                    <---- 451 Access denied CRLF
terminate                           <---- 231 Doe logged out CRLF
                                    Server closes all connections.
```

        [This RFC was put into machine readable form for entry]
         [into the online RFC archives by Via Genie 03/00]