Poste

17454

Received at NIC July 30, 1973

Network Working Group
Request For Comments: # 535
NIC # 17454
Categories: Protocols, FTP
References: RFC 52Ø

Bob Thomas
BBN-TENEX
July 25, 1973

Comments on File Access Protocol

A file access protocol (FAP) of the sort proposed by John Day in RFC 52Ø is a good idea. The following comments suggest improvements (mostly additions) to the protocol described in RFC 52Ø.

1. (Philosophical comment) The intent of both FTP and FAP is to make it possible for a user to remotely access files. In effect, FTP provides means for a user to have (parts of) file activity of the sort typically initiated at the command language level "slaved" across the network to the site where the file resides. In a similar way the intent of FAP is to provide a mechanism which allows activity of the sort typically initiated by programs at the operating system or monitor level to be "slaved" across the network to the site where the file resides. The OPEN, CLOS, SETP, etc. commands of FAP can be viewed as attempts to define "generic" file system monitor calls. The suggestions made below are further attempts to make features typically available to local users also available to remote users via FAP.

2. The OPEN command should allow for a third OPEN mode called A for append. In terms of its action with respect to a file and file pointer, the command

```
OPEN A FOO
```

would be equivalent to the sequence:

```
OPEN W FOO
SETP E
```

The difference would be with respect to access control. Many systems allow a user to control separately write and append access to a file (e.g., on TENEX a user usually sets the protection on his MESSAGE.TXT file such that anyone can append to it but only he can write it). For such systems the append OPEN would succeed in many cases in which the write OPEN would fail. The principle here is that FAP (to as large as degree as is practical) should allow remote users to access files in the same way as local users may.

3. The protocol as proposed allows for the creation of non-sequential files but provides no convenient way for remotely accessing them after they are created. For example if sent to a TENEX server, the sequence:

```
OPEN W FOO      //byte size assumed = 36
SETP B
WRITE 512
SETP 1Ø24
WRITE 512
CLOS
```

would create a file FOO with two pages (on TENEX a page = 512 36 bit words). The two pages would be page #Ø and page #2; because page #1 does not exist the file is said to have

a "hole" in it. Access to FOO via FAP would be difficult unless the remote user knew its (page) structure prior to access. To support remote access to files such as FOO, FAP should have means for a user to determine a file's structure. Consider a value-returning command that returns the value the file pointer should be set to in order to point to the first byte of the next used page (block or record) beyond the current position of the file pointer. With such a command, call it FNUB (Find Next Used Block), the following sequence could be used to retrieve a holey file such as FOO:

```
        OPEN R FILE
        SETP B
  a:    FNUB                  //let x=the value returned
        if x=null
           then CLOS
           else ( SETP x
                  READ 512 //page size=512
                  goto a )
```

This presumes that the remote user knows the block (page) size so that he can properly access the file. One can imagine files having blocks of variable size; perhaps FNUB should return two values: the file pointer position of the next block and the size of that block in bytes.

4. FAP should provide means for a remote user to acquire certain status and "descriptor" information about a given file. The following is a (non-exhaustive) list of information which would be useful to a user remotely accessing TENEX files:

- user's access to file; can he read, write, execute or append the file?
- size information; byte size used in last write access (OPEN W) of the file; file size in bytes (of that size).
- file access dates; date of create, last read, last write.
- on TENEX a user can specify different access control for different pages within the same file; a remote user should be able to acquire such access control information about files (and be able to specify such access control when he creates them).

5. There are many applications in which a remote user would like to access several files simultaneously in much the same way as a local user can. FAP as proposed can not support such multiple file access (of course, the user always has the option of going through an ICP to establish another connection with the server). FAP can be extended in a simple way to support multiple file access by including the notion of a "file handle" which is used to specify which file a given FAP command refers to. When the user does:

OPEN R FOO

the server's response would include a handle for FOO which the user would use in subsequent references to FOO. The handle returned would be a string of the server's choice;

it might be the file's name (FOO), a small integer, etc. Use of a (server chosen) file handle rather than the complete file name enables the server to respond to FAP commands without incurring the overhead of re-parsing the file name for each command. To illustrate, consider the following sequence which opens a file for reading and one for writing, reads 3 bytes from the first file as data, computes using the data and writes a 2 byte result to the second file:

```
OPEN R FOO    //server returns FH as handle
OPEN W MOO    //server returns MH as handle
READ 3 FH     //user reads data
//User does some computation on the 3 bytes
WRIT 2 MH     //user writes the result
CLOS MH
CLOS FH
```

Reasonable defaults could be provided with handles: e.g., a FAP command without a handle refers to the same file as the previous command; etc. (The association of a handle with a file is probably better achieved via a separate FAP command rather than as a side effect of the OPEN command; e.g.,

HNDL FOO)

6. It is important to take local transformations into account (page 3 of RFC 520). However, it is equally important to allow a remote user to suppress local transformations, if he wishes, so that he can access the file as it is stored. This would enable a program that manipulates a file to work equally well whether the file is local (and accessed "directly" via system calls) or remote (and accessed

"indirectly" via system calls that are "trapped" and transformed into FAP commands which are sent to the remote site).