Network Working Group
Request for Comments: 959
Category: Standard

J. Postel, J. Reynolds
ISI

Octobre 1985

French translation by V.G. FREMAUX / Ecole Internationale des Sciences du Traitement de l'Information
(International Graduate School of Computer Sciences)

Remplace RFC: 765 (IEN 149)

# File Transfer Protocol (FTP)

INTERNET
STANDARDS & OFFICIAL DOCUMENTS

Standard 

Informational 

Proposal 

***Note du Traducteur :***

*Le texte suivant est la traduction intégrale de la spécification FTP, telle qu'éditée par les auteurs originaux du protocole, sans ajouts, commentaires, ni omissions. Ce document a valeur normative, selon la procédure courante d'enregistrement des documents au sein du W3C.*

***Statut de ce Mémo***

Ce mémo est la spécification officielle du Protocole de Transfert de Fichiers (FTP). La distribution de ce mémo est illimitée.

Les nouvelles commandes optionnelles suivantes sont inclues dans la présente édition de la spécification:

CDUP (Change to Parent Directory), SMNT (Structure Mount), STOU (Store Unique), RMD (Remove Directory), MKD (Make Directory), PWD (Print Directory), et SYST (System).

Notez que cette spécification est compatible avec la précédente édition.

## 0. TABLE DES MATIERES

# 1. INTRODUCTION

Les objectifs de FTP sont 1) de promouvoir le partage de fichiers (programmes informatiques et/ou données), 2) d'encourager l'utilisation indirecte ou implicite (via des programmes) d'ordinateurs distants, 3) de prémunir l'utilisateur contre les variations de formats de stockage de données entre les différents hôtes, et 4) de transférer les données d'une façon efficace et fiable. FTP, bien que directement utilisable par un utilisateur depuis un terminal, est néanmoins conçu essentiellement pour être utilisé par des programmes.

Cette spécification tente de satisfaire les besoins variés d'utilisateurs de mainframes, minis, et stations personnelles, et TACs, grâce à un protocole au design simple et facile de mise en oeuvre.

Ce document suppose une bonne connaissance du protocole Transmission Control Protocol (TCP) [2] et du protocole Telnet [3]. Ces documents font partie du livre de protocoles ARPA-Internet [1].

# 2. VUE D'ENSEMBLE

Dans cette section, l'historique, la terminologie, et le modèle FTP sont traités. Les termes définis dans cette section sont seulement ce qui ont une signification particulière

pour FTP. Certaines terminologies sont très spécifiques au modèle FTP; certains lecteurs préféreront passer immédiatement à la définition du modèle FTP, quitte à revoir la terminologie par la suite.

### *2.1. HISTORIQUE*

FTP a subi une grande évolution au fil des ans. L'appendice III est une compilation chronologique des RFC se rapportant à FTP. Elle inclue la première proposition de mécanisme de transfert de fichiers de 1971 qui avait été développée pour une application sur les hôtes du M.I.T. (RFC 114), plus des commentaires et discussions dans la RFC 141. La RFC 172 proposait un protocole de niveau utilisateur pour le transfert de fichiers entre ordinateurs (y compris des terminaux IMPs). Une révision de celui-ci (RFC 265), redonnait un état du FTP pour évolution ultérieure, tandis que la RFC 281 suggérait encore d'autres modifications. L'usage d'une transaction "Set Data Type" a été proposée dans la RFC 294 en Janvier 1982. La RFC 354 a rendu les RFC 264 et 265 obsolètes. Le File Transfer Protocol était désormais défini comme un protocole de transfert de fichiers entre des hôtes d'un ARPANET, et dont la fonction première était définie comme le transfert efficace et fiable entre des hôtes pour profiter de l'utilisation d'une capacité de stockage de données distante. La RFC 385 apporte un correctif à certaines erreurs, développe certains points, et ajoute certaines notions au protocole, tandis que la RFC 414 définit le rapport d'état sur le serveur de travail et les "clients" FTP. La RFC 430 de 1973, (parmi d'autres trop nombreuses pour être mentionnées toutes) donnait des commentaires supplémentaires quant à FTP. Finalement, une documentation "officielle" FTP a été publiée sous la référence RFC 454.

Depuis Juillet 1973, des changements considérables sont intervenus, mais la structure globale est restée la même. La RFC 542 a été publiée comme une nouvelle spécification "officielle" pour refléter certains changements. Cependant, de nombreuses implémentations basées sur l'ancienne spécification n'étaient pas remises à jour. En 1974, les RFC 607 et 614 apportent de nouveaux commentaires à propos de FTP. La RFC 624 propose des changements nouveaux et autres modifications mineures. En 1975, la RFC 686 intitulée, "Leaving Well Enough Alone" était une discussion sur les différences entre toutes les anciennes versions de FTP et la dernière en date. La RFC 691 est une révision mineure de la RFC 686, concernant les possibilités d'impression de fichiers.

Motivée par le passage du NCP (Network Communication Protocol) à TCP comme protocole sous-jacent, un phoenix est rené à partir de tous les efforts ci-dessus par la RFC 765 comme une nouvelle spécification de FTP basée sur le protocole réseau TCP.

Cette édition de la spécification FTP est écrite pour corriger quelques erreurs mineures de la RFC 765, tout en étendant les explications de certaines fonctionnalités du protocole, et enfin en ajoutant la définition de quelques commandes supplémentaires. En particulier, les nouvelles commandes optionnelle suivantes sont inclues dans cette édition de la spécification:

CDUP - Change to Parent Directory

SMNT - Structure Mount

STOU - Store Unique

RMD - Remove Directory

MKD - Make Directory

PWD - Print Directory

SYST - System

Cette spécification est compatible avec la version précédente. Un programme implémenté conformément à la précédente spécification devrait naturellement être conforme à la présente.

### *2.2. TERMINOLOGIE*

**ASCII**

Le jeu de caractères ASCII est celui défini par l'ARPA-Internet Protocol Handbook. Pour FTP, les caractères ASCII sont définis comme la moitié de l'ensemble donnée par un codage à huit bits (le bit de poids fort est toujours à 0).

**Contrôle d'accès**

Le contrôle d'accès définit les privilèges utilisateur nécessaires pour utiliser un système, et pour accéder à des fichiers dans ce système. Le contrôle d'accès est nécessaire pour éviter un usage accidentel ou non autorisé de ressources fichiers. Il est dans les prérogatives d'un processus serveur FTP d'invoquer ce contrôle d'accès.

**Tailles d'octets**

Deux tailles d'octets intéressent FTP: la taille des octets logiques du fichier, et la taille utilisée pour la transmission des données. La taille d'octet pour le transfert est toujours de 8 bits. Cette taille de transfert n'est pas nécessairement l'unité d'enregistrement logique du fichier dans le système, ni la taille des unités logiques permettant l'interprétation des structures de données.

**Canal de contrôle**

Le chemin de communication entre le USER-PI et le SERVER-PI pour l'échange de commandes et de réponses à commandes. Cette connexion utilise le protocole Telnet.

**Canal de données**

Une connexion bidirectionnelle (full duplex) sur laquelle les données sont transférées, dans un mode et sous un type particuliers. Les données transférées peuvent être une partie d'un fichier, un fichier entier, ou plusieurs fichiers. Cette connexion s'établit entre un SERVER-DTP et un USER-DTP, ou entre deux SERVER-DTPs.

**Port de données**

Un processus de transfert passif "écoute" sur le **port de données** un ordre de connexion de la part d'un processus de transfert actif émis dans le but d'ouvrir un canal de données.

**DTP**

Le processus de transfert de données DTP (data transfer process) procède à l'établissement et à la gestion de la connexion. Un DTP peut être passif ou actif.

**End-of-Line**

La séquence de fin-de-ligne qui définit la séparation entre deux lignes d'impression. Cette séquence est en général composée d'un Retour Chariot (CR = Carriage Return), suivi d'un saut de ligne (LF = Line Feed).

**EOF**

La condition end-of-file détermine la fin du fichier en cours de transfert.

**EOR**

La condition end-of-record marque la fin d'un enregistrement de données en cours de transfert.

**correction d'erreur**

Une procédure qui permet à un utilisateur de se récupérer suite à certaines erreurs telles qu'une faute du système serveur ou du processus de transfert lui-même. Pour FTP, la correction d'erreurs nécessitera un redémarrage de la transmission d'un fichier à partir d'un point de contrôle donné.

**Commandes FTP**

Un ensemble de commandes comprenant le contrôle des informations transitant entre le USER-FTP et le SERVER-FTP.

**file**

Une suite ordonnée (séquentielle) de données informatiques (y compris des programmes), d'une longueur arbitraire, et définies parfaitement par un "chemin d'accès".

**mode**

Le mode dans lequel les données doivent être transmises. Le mode définit le format des données pendant la tansmission, y compris les conditions EOR et EOF. Les modes de transfert définis par FTP sont décrits dans la section traitant des Modes de Transmission.

**NVT**

Le Network Virtual Terminal défini par le protocole Telnet.

**NVFS**

Le Network Virtual File System. Un concept qui définit un système de fichiers standard vu à travers le réseau utilisant des conventions standardisées de commandes et de syntaxe de noms de chemins d'accès.

**page**

Un fichier peut être structuré comme un ensemble de parties indépendantes appelées pages. FTP supporte la transmission de fichiers discontinus comme une suite de pages indexées indépendantes.

**Chemin d'accès**

Le chemin d'accès est défini comme la chaîne de caractères qui doit être présentée par un utilisateur à un système de fichier pour localiser une ressource. Le chemin d'accès contient normalement une indication de l'unité logique et/ou des noms de répertoires, et enfin un nom de fichier. FTP ne spécifie aucune convention particulière pour le chemin d'accès. Chaque utilisateur devra se conformer aux conventions utilisées sur les systèmes de fichiers impliqués dans le transfert.

**PI**

Le Protocol Interpreter (interpréteur de protocole). Les côtés serveur (SERVER) et utilisateur (USER) d'un protocole ont des "rôles" distincts implémentés respectivement dans un SERVER-PI et un USER-PI.

**enregistrement**

Un fichier à accès séquentiel peut être structuré comme un certain nombre de portions contiguës appelés enregistrements. Les structures en Enregistrements sont supportés par FTP bien qu'un fichier n'ait nul besoin d'être organisé de cette façon.

**réponse**

Une réponse est un acquittement ou une dénégation envoyée par un serveur à l'utilisateur via la connexion de contrôle en réponse à une commande FTP. La forme générale d'une réponse est un code de résultat (pouvant être un code d'erreur) suivi d'une chaîne de caractères. Les codes sont à destination d'agents logiciels, le texte est plus naturellement destiné à des utilisateurs humains.

**SERVER-DTP**

Le processus qui transmet les données, dans son état "actif" normal, établit le canal de données sur le port "en écoute". Il établit des paramètres pour le transfert et le stockage, et transfère les données sur commande de son PI. Le DTP peut entrer dans un état "passif" pour attendre, plutôt qu'initier une communication.

**Processus serveur FTP**

Un processus ou ensemble de processus qui prennent en charge la fonction de transfert de fichiers en coopération avec un processus USER-FTP et, certainement un

autre serveur. La fonction rassemble un interpréteur de protocole (PI) couplé à un processus de transfert de données (DTP).

**SERVER-PI**

L'interpréteur de protocole serveur "écoute" sur le Port L une communication arrivant d'un USER-PI et établit la connexion pour le canal de contrôle. Il reçoit par celui-ci les commandes FTP de l'USER-PI, y répond, et pilote le SERVER-DTP.

**type**

Le type de représentation de données utilisé pour la transmission et le stockage. Le type implique certaines différences entre le temps d'enregistrement et le temps de transfert. Les types de représentation de données définis dans FTP sont décrits dans la Section traitant de l'établissement des canaux de données.

**Utilisateur (USER)**

Une personne ou un processus sous contrôle d'une personne désirant obtenir des fichiers distants par transfert. L'utilisateur "humain" peut directement agir en interactivité avec un processus SERVER-FTP, mais le passage par un processus USER-FTP est conseillé dans la mesure où le protocole FTP a été conçu sur un concept d'automate.

**USER-DTP**

Le processus de transfert de données "écoute" le port de données en attendant la connexion à un processus SERVER-FTP. Si deux serveurs transfèrent des données entre eux, le processus USER-DTP est inactif.

**Processus USER-FTP**

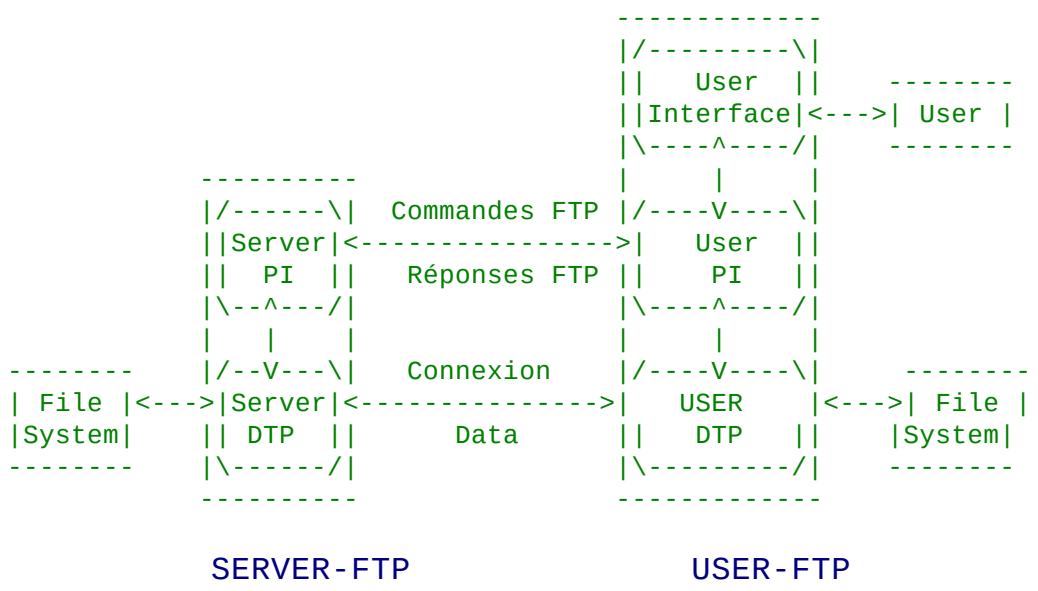
Un ensemble de processus et de fonctions incluant un interpréteur de protocole, un processus de transfert de données et une interface utilisateur par laquelle la fonction de transfert de fichier peut être effectuée en coopération avec un ou plusieurs processus SERVER-FTP. L'interface utilisateur met à disposition de l'utilisateur un langage local de commande-réponse.

**USER-PI**

L'interpréteur de protocole utilisateur instaure le canal de contrôle via son port U avec le processus SERVER-FTP, émet des commandes FTP, et gouverne le USER-DTP si ce dernier est impliqué dans le processus de transfert.

### *2.3. LE MODELE FTP*

Avec les définitions ci-dessus à l'esprit, le modèle suivant (montré en Figure 1) peut être explicité pour la mise en oeuvre d'un service FTP.




NOTES: 1. La connexion de données peut être utilisée dans les deux directions.
2. Il n'est pas nécessaire que le canal de données soit maintenue en permanence.

*Figure 1 Modèle d'usage de FTP*

Dans le modèle décrit en Figure 1, l'interpréteur de protocole utilisateur (USER-PI) instaure le canal de contrôle. Ce circuit de communication utilise le protocole Telnet. A l'instauration de cette connexion, des commandes FTP standard sont générées par le USER-PI et transmises au processus serveur via le canal de contrôle. (L'utilisateur pourra néanmoins établir une liaison de contrôle directe avec le SERVER-FTP, à partir d'un terminal TAC par exemple, et générer les commandes standard indépendamment, en se substituant au processus USER-FTP). Des réponses standardisées sont émises en retour par le SERVER-PI au USER-PI via le canal de contrôle alors établie.

Les commandes FTP spécifient les paramètres du canal de données (port de données, mode de transfert, type pour la représentation, et structure des données) ainsi que la nature du fonctionnement des systèmes de fichiers (enregistrement, lecture, ajout, suppression, etc.). Le USER-DTP ou son délégué se mettra en "écoute" sur le port de données spécifié, et le serveur instaurera le canal de données et effectuera le transfert de fichiers selon les paramètres spécifiés. Il doit être noté que le port de données n'est pas nécessairement sur le même hôte que celui qui a émis les premières commandes FTP par son canal de contrôle, bien que l'utilisateur ou le USER-FTP doive continuer à assurer "l'écoute" sur le port spécifié. Il doit être ici signalé en outre que le canal de données mis en place peut servir simultanément à la lecture et à l'écriture de données.

Une autre situation peut consister en un utilisateur qui souhaite transférer des fichiers entre deux hôtes, les deux étant des hôtes distants différents de celui de

l'utilisateur. L'utilisateur établit alors un canal de contrôle vers chacun des deux serveurs et utilise ces canaux pour créer un canal de données entre ces deux hôtes.

De cette façon, les informations de contrôle passent par le USER-PI bien que les données soient transmis ente deux processus serveurs de transfert. Ce qui suit est un modèle de cette interaction entre serveurs.

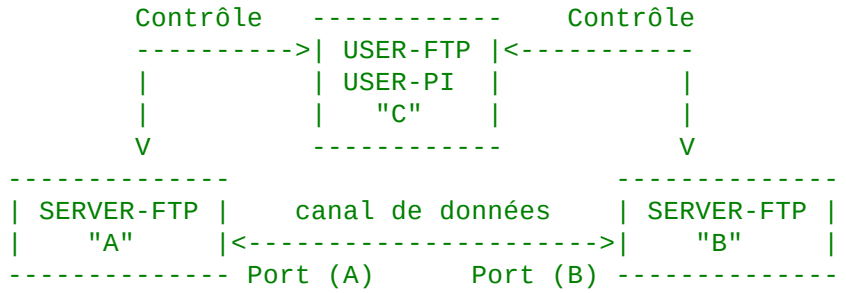



*Figure 2*

Le protocole demande à ce que les canaux de contrôle soient ouvert tant que dure le transfert de données. Il est de la responsabilité de l'utilisateur de demander la fermeture des canaux de contrôle lorsque l'utilisation du service FTP est terminée. C'est néanmoins le processus serveur qui prend en charge la rupture. Le serveur peut arrêter un transfert de données si le canal de contrôle est coupé sans commande préalable.

***Relations entre FTP et Telnet:***

FTP s'appuie sur le protocole Telnet pour le dialogue du canal de contrôle. Ceci est effectif en deux sens: premièrement, le USER-PI ou le SERVER-PI devront suivre les règles du protocole Telnet directement dans leur propres procédures; ou bien, le USER-PI ou le SERVER-PI peuvent faire appel à un module Telnet existant et disponible dans le système d'exploitation.

La facilité d'implémentation, les principes de réutilisabilité, et la programmation modulaire font pencher en faveur de la deuxième solution. L'efficacité et l'indépendance vis à vis de la plate-forme sont des argument en faveur de la première. En pratique, FTP n'utilise qu'un tout petit sous ensemble du protocole Telnet, et de ce fait, la première approche n'induit pas un travail de programmation insurmontable.

## 3. FONCTIONS DE TRANSFERT DE DONNEES

Seul le canal de données permet le transfert effectif des fichiers. La canal de contrôle n'est utilisé que pour le contrôle des commandes, qui indiquent les fonctions qui doivent être exécutées, ainsi que les réponses à ces commandes (voir la Section traitant des Réponses FTP). Plusieurs commandes concernent le transfert de données entre hôtes. Ces commandes de transfert incluent la commande MODE qui spécifie comment les bits de données doivent être transmis, ainsi que les commandes STRUcture et TYPE, qui sont utilisées pour définir la manière avec laquelle sont représentées les données. La transmission et la représentation sont des notions indépendantes. Cependant le mode de transmission "Stream" reste dépendant des

attributs de structure des fichiers et si le mode de transmission "Compressed" est utilisé, la nature des octets de remplissage dépendra de la représentation des données utilisée.

### *3.1. REPRESENTATION DES DONNEES ET STOCKAGE*

Les données sont transférées à partir d'un espace de stockage dans l'hôte émetteur vers l'espace de stockage de l'hôte récepteur. Il est souvent nécessaire d'effectuer certaines transformations sur les données du fait de la différence de la représentation de ces dernières dans deux systèmes de nature différente. par exemple, le format NVT-ASCII est stocké sous diverses représentations selon le système. Les DEC TOPS-20 enregistrent généralement le format NVT-ASCII sous la forme de cinq caractères ASCII codées sur 7 bits, dans un mot de 36-bit calé sur la gauche. Les mainframes IBM enregistre ce même format sous forme de codes EBCDIC sur 8 bits. Le système Multics stocke le format NVT-ASCII sous la forme de quatre caractères sur 9-bits dans un mot de 36-bits. Il est souhaitable de convertir les caractères entre les diverses représentation du format NVT-ASCII lorsque des transmissions sont effectuées entre systèmes distincts, en passant par une représentation standard. Les sites émetteurs et récepteurs devront effectuer les transformations nécessaires entre la représentation standard commune et leur propre représentation interne.

Un autre problème de représentation apparaît lors du transfert de données binaires (codes non assimilables à du texte) entre deux systèmes travaillant avec des largeurs de mots distinctes. La façon dont l'émetteur envoie les données n'est pas toujours exprimée explicitement, pas plus que la façon dont le récepteur les stocke. Par exemple, lors de la transmission de "mots" de 32-bits à partir d'un système 32-bits vers un systèmes fonctionnant en 36-bits, il peut être souhaitable (pour des raisons de performance) de stocker les mots de 32-bits calés à droite du mot de 36-bits du système récepteur. Dans tous les cas, l'utilisateur doit avoir accès à l'option qui lui permettra de spécifier la représentation des données, et les transformations nécessaires. Il doit être noté que FTP n'admet qu'un nombre limité de formats de données. Les transformations en dehors du contexte limité proposé par FTP devront être prises en charge par l'utilisateur.

#### 3.1.1. TYPES DE DONNEES

Les représentations de données sont gérées dans FTP par la spécification par l'utilisateur d'un type. Ce type peut être implicite (comme pour l'ASCII ou l'EBCDIC) ou explicite (comme le type Local), et définit une taille de mot dont l'interprétation correspond à celle de la "taille de mot logique". Notez que ceci n'a rien à voir avec la taille du mot utilisé dans la transmission dans le canal de données, appelée la "taille de transfert", la confusion entre les deux notions devant être soigneusement évitée. Par exemple, le format NVT-ASCII a une taille logique de 8 bits. Si le type est le type Local, alors la commande TYPE aura un deuxième paramètre obligatoire spécifiant cette taille logique. La taille de transfert est toujours égale à 8 bits.

#### 3.1.1.1. TYPE ASCII

C'est le type par défaut et doit être reconnu par toutes les implémentations FTP. Il est à l'origine mis en place pour le transfert de fichiers texte, sauf lorsque les deux hôtes considéreront que le type EBCDIC convient mieux.

L'émetteur convertit les données depuis la représentation interne des caractères vers le format 8-bit NVT-ASCII standardisé (voir les spécifications Telnet). Le récepteur convertira cette représentation standard en sa propre représentation interne.

Conformément au standard NVT, la séquence <CRLF> doit être utilisée à chaque fois que nécessaire pour marquer une fin de ligne de texte. (Voir la discussion à propos des structures de fichiers à la fin de la Section traitant des Représentations de données et stockage). Le fait d'utiliser la représentation standard NVT-ASCII en 8 bits signifie que les données doivent être interprétées selon des "mots" de 8-bits. Les valeurs du paramètre Format pour les types ASCII et EBCDIC sont détaillées ci-après.

#### 3.1.1.2. TYPE EBCDIC

Ce type est destiné à des transferts plus efficaces entre deux hôtes qui admettent l'EBCDIC comme standard de représentation interne des caractères de texte.

Pour la transmission, les données sont représentées comme des codes EBCDIC sous 8-bits. Le codage des caractères est la seule différence qui distingue les spécifications des types EBCDIC et de l'ASCII.

La fin de ligne (EOL équivalent à la séquence CRLF) (par opposition à la fin d'enregistrement (EOR)—voir la discussion sur les structures) sera rarement utilisée avec le type EBCDIC pour des raisons de reconnaissances de structure, mais lorsqu'une telle information est nécessaire, le caractère <NL> pourra être utilisé.

#### 3.1.1.3. TYPE IMAGE

Les données sont transmises comme un champ de bits continu qui, pour le transfert, sont "empaquetés" dans des structures de transfert de 8-bits. Le site récepteur doit quant à lui enregistrer les données comme un champ continu de bits. La structure du système de stockage nécessite parfois l'utilisation de bits de "bourrage" pour le fichier (ou pour chaque enregistrement, dans le cas d'un fichier structuré sur une base d'enregistrements logiques) établissant ainsi un "calage" des données sur une frontière conventionnelle (octet, mot ou bloc). Ce bourrage doit toujours être fait par des bits nuls, peut intervenir à la fin d'un fichier (ou à la fin de chaque enregistrement) et il doit exister un moyen de les identifier afin qu'ils puissent être éliminés lorsque le fichier est récupéré. La transformation de bourrage devra faire l'objet d'une large et claire documentation pour permettre à tout utilisateur d'implémenter les traitements nécessaires à la reconstitution du fichier original dans le site récepteur.

Le type image est destiné à un transfert et un enregistrement optimal de fichiers binaires. Il est recommandé que ce type soit reconnu par toutes les implémentations FTP.

#### 3.1.1.4. TYPE LOCAL

Les données sont transférées par mots logiques dont la taille est nécessairement spécifiée par un second paramètre obligatoire, "Byte size". La valeur du paramètre "Byte size" doit être un entier décimal; il n'existe pas de valeur par défaut. La taille de mot logique n'est pas nécessairement la même que celle du mot de transfert. Si les deux tailles sont différentes, alors les mots logiques devront être empaquetés selon une trame continue de bits, indépendamment des limites formées par le mot de transfert et avec le bourrage nécessaire ajouté à la fin.

Lorsque les données sont reçues sur l'hôte récepteur, elles seront transformées selon la taille des mots logiques du fichier transféré et la taille de la représentation interne du récepteur. Cette transformation doit être réversible (c-à-d, un fichier identique doit pouvoir être récupéré dans l'autre sens avec les mêmes paramètres) et devra faire l'objet d'une documentation précise et complète de la part des implémenteurs FTP.

Par exemple, un utilisateur envoyant des nombres à virgule flottante en 36-bits vers un hôte travaillant en 32-bits pourrait envoyer ces données sous le type Local selon une taille Locale de 36. L'hôte récepteur pourrait par exemple récupérer ces mots logiques et les enregistrer d'une façon à ce qu'ils puissent être manipulés facilement; dans notre exemple, une solution consiste à stocker les mots de 36-bits dans un double mot de 64-bits.

Autre exemple : le cas d'une paire d'hôtes travaillant sous 36-bits pourraient se communiquer des données en utilisant le TYPE L 36. Les données seraient alors transmises empaquetées dans le format 8-bits de la transmission, 9 octets transmis étant nécessaires pour transférer deux "mots" entre deux tels systèmes.

#### 3.1.1.5. CONTROLE DE FORMAT

Les types ASCII et EBCDIC prennent un second paramètre (optionnel); il indique quel type de contrôle de format vertical , s'il existe, est associé à un fichier. Les types de représentation de données suivantes sont définis dans FTP:

Un fichier caractères peut être transféré vers un hôte dans l'un des trois buts suivants : pour impression, pour stockage et récupération ultérieure, ou pour traitement. Si un fichier est envoyé pour impression, l'hôte récepteur doit connaître comment le contrôle de format vertical format est représenté. Dans le second cas, il doit être possible d'enregistrer un fichier pour usage ultérieur dans sa forme originale. Enfin, il doit être possible de déplacer un fichier d'un hôte vers un autre, et de traiter ce fichier sur l'hôte récepteur sans ennui. Un format ASCII ou EBCDIC élémentaire ne satisfait pas à ces conditions. De ce fait, un second paramètre a été adjoint au paramètre de type, pour coder trois situations possibles :

##### 3.1.1.5.1. NON PRINT

C'est le format par défaut à utiliser si le second paramètre (format) est omis. Le format NON-PRINT doit être accepté par toutes les implémentations de FTP.

Le fichier ne contient pas nécessairement des informations de contrôle vertical. Si un tel fichier est passé à un processus d'impression, ce dernier devra prendre des

valeurs standard pour les espaces et les marges. Ce format sera usuellement utilisé pour des fichiers destinés à du traitement de données ou à être juste stockés.

3.1.1.5.2. TELNET FORMAT CONTROLS

Le fichier contient des codes ASCII/EBCDIC de contrôle de format vertical (c-à-d., <CR>, <LF>, <NL>, <VT>, <FF>) qu'un processus d'impression peut immédiatement interpréter. <CRLF>, dans cet ordre précis, signale une fin de ligne.

3.1.1.5.2. CARRIAGE CONTROL (ASA)

Le fichier contient des caractères de contrôle de format vertical conformes à l'ASA (FORTRAN). (Voir RFC 740 Appendice C; et "Communications of the ACM, Vol. 7, No. 10, p. 606, October 1964".) Dans une ligne, ou un enregistrement au format conforme au standard ASA, le premier caractère ne doit pas être imprimé. Au lieu de cela, il doit être utilisé pour déterminer le mouvement vertical du papier à effectuer avant que l'impression du reste de l'enregistrement ne soit effectué.

Le standard ASA spécifie les caractères de contrôle suivants :

| *Caractère* | *Espacement vertical* |
|---|---|
| **Espace** | Avance le papier d'une ligne |
| **0** | Avance le papier de deux lignes |
| **1** | Avance le papier en début de la page suivante |
| **+** | Pas de mouvement (surimpression) |

Il doit exister un moyen simple pour un processus d'impression de détecter la fin d'une entité structurale. Si un fichier est enregistré selon une structure d'enregistrement (voir ci-dessous), il n'y a aucun problème; les enregistrements seront explicitement marqués pendant le transfert et l'enregistrement. Si le fichier n'a aucune structure d'enregistrement sous-jacente, la séquence de fin de ligne <CRLF> est utilisée pour séparer les lignes d'impression, bien que l'effet produit par ces deux caractères soit masqué par la signification des contrôles ASA.

3.1.2. STRUCTURES DE DONNEES

En plus des différents types de représentation de données, FTP permet que la structure d'un fichier soit explicitée. Trois structures de fichiers sont connues de FTP:

*Structure fichier,* dans laquelle le fichier est considéré comme une séquence continue d'octets contigus, sans structure sous-jacente induite.

*Structure-enregistrement,* dans laquelle un fichier peut être vu comme une séquence d'enregistrements,

*Structure-page,* dans laquelle le fichier peut être considéré comme une suite de pages indépendantes indexées.

La "structure-fichier" est la structure par défaut et doit être considérée si la commande STRUcture n'a pas été utilisée, bien que les deux structures "fichier" et "enregistrement" dussent être acceptées pour les fichiers "texte" (c-à-d., fichiers affichant un TYPE ASCII ou EBCDIC) par toutes les implémentations FTP. La structure d'un fichier affectera à la fois la façon de transmettre le fichier (voir la Section traitant du Mode de Transmission) et l'interprétation de l'enregistrement sur le support de stockage.

La structure "naturelle" d'un fichier dépend de l'hôte qui l'enregistre. Du code source sera généralement enregistré sur un mainframe IBM comme une suite d'enregistrements de longueur fixe, et au contraire comme un flux de caractères séparé en lignes par une séquence <CRLF> par exemple, sur un DEC TOPS-20. Si le transfert de fichiers entre des sites aussi différents s'avère utile, il doit exister un moyen de différencier les stratégies de codage de chaque côté de la transaction.

Entre des sites naturellement orientés vers une structure "fichier" et d'autres utilisant naturellement une structure "enregistrements", on pourra rencontrer des problèmes à transférer un fichier basé sur une des deux structures vers un système s'appuyant sur l'autre. Si un fichier texte organisé en "enregistrements" est envoyé vers un hôte naturellement orienté "fichier", alors ce dernier devra appliquer une transformation interne pour l'enregistrer. Cette transformation est évidemment utile, mais doit être de plus totalement réversible pour assurer une récupération "à l'identique".

Dans le cas inverse de fichiers de type "fichier", vers un hôte travaillant en structures "enregistrement", se pose le problème de savoir quel sera le critère utilisé pour recomposer le fichier selon une structure d'enregistrements. Si cette division est nécessaire, l'implémentation FTP devrait utiliser la séquence fin-de-ligne, <CRLF> pour l'ASCII, ou <NL> pour les fichiers texte EBCDIC, comme délimiteur d'enregistrement. Si une implémentation FTP adopte cette technique, elle doit être prête à pouvoir procéder à la transformation inverse au cas où le fichier devrait être rapatrié vers son support original de type "fichier".

#### 3.1.2.1. STRUCTURE FICHIER

La structure "fichier" est à considérer par défaut si la commande STRUcture n'est pas employée.

Dans une structure-fichier, il n'y a en fait aucune structure sous-jacente et le fichier doit être considéré comme une suite continue de caractères.

### 3.1.2.2. STRUCTURE ENREGISTREMENT

La structure-enregistrement doit être acceptée pour tout fichier "texte" (c-à-d., fichiers affichant un TYPE ASCII ou EBCDIC) par toutes les implémentations FTP.

Le fichier est alors reconnu comme une suite ordonnée d'enregistrements successifs.

### 3.1.2.3. STRUCTURE PAGINEE

Pour transmettre des fichiers discontinus, FTP définit une structure en pages. Les fichiers de ce type sont aussi connus comme des "fichiers à accès aléatoire" par

opposition aux "fichiers à accès séquentiel". Dans ces fichiers, il existe souvent un certain nombre d'informations annexes, associées au fichier lui même (ex., un descripteur du fichier), ou à l'une de ses parties (ex., des contrôles d'accès aux différentes pages), ou les deux. Pour FTP, chaque section séquentielle d'un tel fichier est appelée page.

Afin d'exploiter des tailles et des attributs de page différents, chaque page est envoyée avec une en-tête. L'en-tête contient une sélection des paramètres suivants:

| | |
|---|---|
| Header Length<br>*(Longueur d'en-tête)* | Le nombre d'octets logiques dans l'en-tête y compris lui-même. La longueur minimale d'en-tête est de 4. |
| Page Index<br>*(Index de page)* | L'index de cette section du fichier (numéro de page logique). Ceci n'est pas le numéro de page transmise, mais plutôt l'index qui permet d'identifier cette page. |
| Data Length<br>*(Longueur des données)* | Le nombre d'octets logiques dans la page. La longueur de page peut être nulle. |
| Page Type<br>*(Type de page)* | Le type de page dont il s'agit. Sont définis les types qui suivent : |
| 0 = Last Page<br>*(Dernière page)* | Utilisé pour indiquer la fin d'une transmission structurée en pages. La longueur d'en-tête de cette page est 4, et la longueur de page nécessairement 0. |
| 1 = Simple Page<br>*(Page simple)* | Type d'une page normale du fichier ne disposant pas d'information de contrôle hiérarchique. La longueur d'en-tête doit être de 4. |
| 2 = Descriptor Page<br>*(Descripteur)* | Type utilisé pour transmettre en un bloc toute la description externe du fichier. |
| 3 = Access Controlled Page<br>*(Page à accès contrôle)* | Ce type inclut un paramètre supplémentaire destiné à l'accès à des pages organisées selon une structure hiérarchique à plusieurs niveaux. L'en-tête est de longueur 5. |
| Champs optionnels | Des champs optionnels peuvent être ajoutés pour fournir une information spécifique sur la page elle-même, par exemple un contrôle d'accès individuel. |

Tous les champs sont de longueur égale à un octet logique. La taille de l'octet logique est défini par le paramétrage de la commande TYPE. Voir l'Appendice I pour plus de détails.

*Note d'avertissement concernant les paramètres :* un fichier doit être téléchargé, enregistré et récupéré avec les mêmes paramètres si l'on souhaite récupérer une version identique à l'original. A l'inverse, les implémentations de FTP doivent renvoyer un fichier identique à l'original si les paramètres utilisés pour l'enregistrement et la récupération du fichier sont identiques.

### *3.2. ETABLISSEMENT DU CANAL DE DONNEES*

Le mécanisme de transfert de données consiste en l'établissement d'un canal de données entre les ports appropriés et de ce fait en le choix des paramètres de transfert. Le USER et SERVER-DTP disposent tous deux d'un port de données par défaut. Le port "données" par défaut du processus utilisateur est identique à celui utilisé pour le contrôle de la connexion (c-à-d., U). Le port "données" par défaut du processus serveur est le port adjacent à celui utilisé pour le contrôle de la connexion (c-à-d., L-1).

La taille de l'octet transféré est toujours de 8-bits. Cette taille n'a de signification que pendant le processus effectif de transfert des données; elle ne présume en rien de la taille des unités logiques nécessaires pour représenter les données à l'intérieur du système.

Le processus de transfert de données à l'état passif (ceci peut être un USER-DTP ou un deuxième SERVER-DTP) devra "écouter" son port de données avant de pouvoir émettre une commande de requête de transfert. La commande FTP de requête de transfert détermine le sens du transfert de données. Le serveur, sur réception de la requête, établira la connexion au port "données". Lorsque cette dernière est établie, le transfert de données débute entre les deux DTP, et le SERVEUR-PI émet une confirmation à destination du USER-PI.

Toute implémentation FTP doit accepter l'utilisation des ports par défaut, et seul le USER-PI peut invoquer une migration de la connexion vers des ports non standards.

Le processus utilisateur peut demander l'usage d'un autre port "données" par l'intermédiaire de la commande PORT. Par exemple, un utilisateur demande l'impression d'un fichier sur une imprimante en ligne TAC lequel fichier doit être récupéré depuis un troisième hôte. Dans le dernier cas, le USER-PI établit un canal de contrôle avec les deux SERVER-PI. Il est alors demandé à un serveur (par une commande FTP) "d'écouter" une connexion qu'une troisième entité va initier. Le USER-PI émet à destination d'un des SERVER-PI une commande PORT indiquant le port "données" de l'autre connexion. Enfin, il est envoyé aux deux serveurs les commandes de transfert appropriées. La séquence exacte de commandes et de réponses envoyées entre le contrôleur de l'utilisateur et les serveurs est définie dans la Section traitant des Réponses FTP.

En général, il est de la responsabilité des serveurs de maintenir le canal de données actif—de l'initialiser et de le clore. L'exception à cette règle est lorsque le USER-DTP envoie des données dans un mode qui implique que la fin de fichier (EOF) correspond à la fermeture de la transmission. Le serveur DOIT fermer le canal de données sous les conditions suivantes :

1. Le serveur à terminé la transmission de données dans un mode ou la fin de fichier est signalée par une fermeture du canal.

2. Le serveur reçoit une commande ABORT de l'utilisateur.

3. La spécification du port "données" est changée par une commande de l'utilisateur.

4. Le canal de contrôle est fermé par une procédure normale ou pour toute autre raison.

5. Une condition d'erreur irrécupérable est apparue.

Dans tous les autres cas la fermeture est une prérogative du serveur, l'exercice de la quelle doit être signalée au processus utilisateur par un code de réponse 250 ou 226 seulement.

### *3.3. GESTION DU CANAL DE DONNEES*

Ports de données standard : Toute implémentation FTP doit accepter l'usage des ports de données standard, seul un USER-PI pouvant initialiser un canal sur un port autre que standard.

Négociation des ports autres que par défaut : Le USER-PI peut spécifier un port de données non standard à "viser" par le serveur via la commande PORT. Le USER-PI peut demander au serveur de s'identifier au serveur "cible" exprimé par ce port non standard via la commande PASV. La connexion étant définie comme une paire d'adresse, ces deux actions sont suffisantes pour obtenir à chaque fois un canal de données différent, bien qu'il soit admis de pouvoir déclencher deux fois ces commandes pour raccorder deux ports non standard à chaque extrémité d'un canal de données.

Réutilisation du canal de données : Lorsque le mode de transfert en "flux" est utilisé, la fin de fichier est indiquée implicitement par une fermeture du canal. Ceci pose un problème évident lorsque plusieurs fichiers doivent être transférés au cours de la même session, dans la mesure où TCP doit "bloquer" la connexion qui vient d'être utilisée pendant un certain temps fixé pour des raisons de fiabilité. De ce fait, une connexion ouverte sous ce mode ne peut pas être réutilisée immédiatement.

On donnera deux solutions à ce problème. La première est de négocier un autre canal sur des ports non standard. La première est de changer le mode de transfert.

Commentaire sur les modes de transfert. Le mode de transfert en "flux" est par nature non fiable, dans la mesure où il est impossible de déterminer si un canal est fermé normalement ou non. Les autres modes de transfert (Bloc, Compressé) ne ferment pas le canal après transmission du fichier. Le niveau d'encodage de FTP est suffisant pour que le canal puisse être "surveillé" et que la fin de fichier puisse être détectée. Ces modes sont donc tout à fait exploitables pour la transmission de multiples fichiers.

### *3.4. MODES DE TRANSMISSION*

La considération suivante à prendre en compte pour transférer des fichiers est le choix d'un mode de transmission. FTP définit trois modes : un qui formate les données est permet de recommencer la transmission si nécessaire; une qui compresse en plus les données pour un transfert plus efficace; et un dernier mode qui laisse passer les données avec le moins d'encodage possible. Dans ce dernier cas, le mode interagit avec les attributs de structure pour déterminer le type de traitement. En mode compressé, le type de représentation détermine essentiellement la nature du bourrage.

Tous les transferts de données doivent s'achever par la transmission d'une séquence de fin-de-fichier (EOF), la quelle peut être explicite, ou implicitement déduite

de la fermeture du canal. Pour les fichiers de structure de type "enregistrement", tous les marqueurs de fin d'enregistrement (EOR) sont explicites, y compris le dernier. Pour les fichiers transmis selon une structure de pages, la page de type "last-page" sera utilisée pour marquer la fin de la transmission.

*NOTE: Dans le reste de cette section, octet signifiera "l'octet de transfert" sauf mention contraire explicite.*

Dans le but d'obtenir un transfert standardisé, l'hôte émetteur devra traduire sa représentation interne d'une fin de fichier ou fin d'enregistrement dans la représentation préconisée par le protocole pour le mode de transfert et la structure de fichier donnés, l'hôte récepteur effectuant la transcription duale vers sa propre représentation interne. Le champ de comptage d'enregistrements d'un mainframe IBM peut ne pas être reconnu par un autre hôte, l'information de fin d'enregistrement devant alors être transféré comme un code de contrôle à deux octets en mode "flux" ou par marquage de bits dans les descripteurs des modes Bloc ou Compressé. La fin de ligne dans un fichier ASCII ou EBCDIC sans structure d'enregistrement devrait être indiqué par une séquence <CRLF> ou <NL>. Comme ces transformation impliquent un travail supplémentaire dans les hôtes, des systèmes identiques ou similaires s'échangeant des fichiers préféreront utiliser un transfert binaire dans un mode de type "flux".

Les modes de transmission suivants sont reconnus par FTP :

### 3.4.1. MODE "FLUX"

Les données sont transmises comme un flux d'octets. Il n'y a dans ce cas aucune restriction sur la représentation des données utilisée; Des structures-enregistrement sont autorisées.

Dans un fichier d'enregistrements, les séquences EOR et EOF seront toutes deux marquées par un œode de contrôle à deux octets. Le premier octet vaudra 0xFF, le caractère d'échappement. Le second octet aura sont bit de poids faible à 1 et des 0 ailleurs pour la marque EOR (le second bit à 1 pour la marque EOF); en somme, l'octet aura la valeur 1 pour l'EOR et 2 pour l'EOF. EOR et EOF peuvent être marqué simultanément dans la dernière séquence en marquant les deux bits dans le même octet (donc, une valeur 3 pour le dernier enregistrement). Si un octet de données devait avoir la valeur OxFF, il devra être répété dans le second octet du code de contrôle.

Si la structure est de type "fichier", la séquence EOF sera implicitement marquée par la fermeture du canal. Tous les octets transmis sont donc des octets de données.

### 3.4.2. MODE BLOC

Le fichier est transmis comme une suite de blocs de données précédés d'un ou plusieurs octets d'en-tête. L'en-tête contient un champ de comptage de blocs, et un code de description. Le champ de comptage indique la longueur totale du bloc de données en octets, et indique donc le début du bloc suivant (il n'y a pas de bits de bourrage). Le code de description indique : le dernier bloc du fichier (EOF) le dernier bloc de l'enregistrement (EOR), le marqueur de reprise (voir la Section traitant de la Récupération d'Erreurs et Reprise de Transmission) ou de données suspectes (c-à-d. qu'il est possible que les données transférées soient erronées et non fiables). Ce

dernier code N'EST PAS destiné à implémenter une fonction de contrôle d'erreur sous FTP. Il est motivé par la demande de certains sites d'échanger des classes particulières de données (ex., données sismiques ou météorologiques) en dépit d'erreurs locales qui peuvent survenir (telles que des erreurs de lecture sur des supports magnétiques), pour indiquer que certaines données transmises peuvent être suspectes pour des raisons autres que la transmission. Des structures-enregistrement sont admises dans ce mode, et toute forme de représentation de données peut être utilisée.

L'en-tête consiste en trois octets. Sur ces 24 bits d'information d'en-tête, les 16 bits de poids faible représentent le compte d'octets, les 8 bits de poids fort donnent le code de description selon les définitions ci-dessous.

***Bloc d'en-tête***

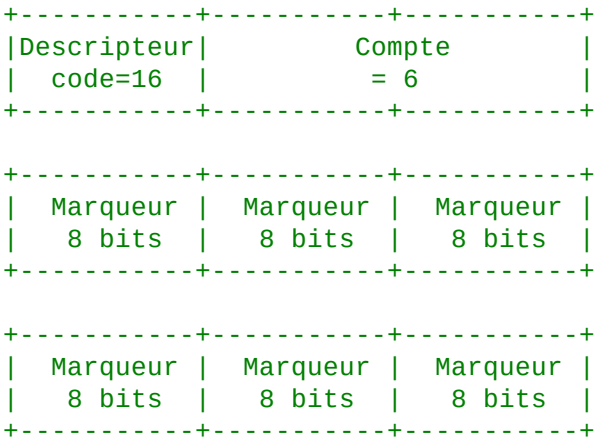

Le descripteur est formé de bits indicateurs. Quatre codes sont actuellement reconnus, dont le nombre représente la valeur décimale du masque.

***Codes de description***

| *Valeur* | *Description* |
|---|---|
| **128** | Le bloc est le dernier d'un enregistrement (EOR en fin de bloc) |
| **64** | Le bloc est le dernier de la transmission (EOF en fin de bloc) |
| **32** | Erreurs probables dans les données de ce bloc |
| **16** | Le bloc de données est le premier d'une reprise |

Grâce à cet encodage, plusieurs situations simultanées peuvent être codées dans un seul bloc. Autant de bits du descripteur que nécessaire peuvent être marqués.

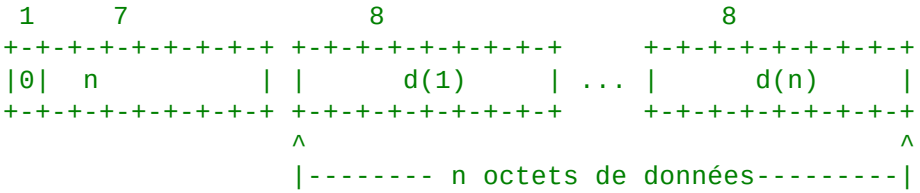
Le marqueur de reprise est émis comme des données d'un multiple entier d'octets de 8-bits représentant des caractères imprimables selon le langage utilisé sur le canal de contrôle (ex., par défaut--NVT-ASCII). <SP> (Espace, dans le langage approprié) ne doit JAMAIS être employé dans un marqueur de reprise.
Par exemple, pour transmettre un marqueur de reprise de six caractères, la séquence suivante serait émise :




### 3.4.3. MODE COMPRESSE

Trois classes d'informations doivent être envoyées : des données "littérales", envoyées comme des chaînes d'octets; des données compressées, consistant en des octets "répliqués" ou des octets de bourrage; et des informations de contrôle, émis selon des séquences d'échappement à deux octets. Si n>0 octets (jusqu'à 127) littéraux sont émis, ces n octets doivent être précédés d'un octet dont le bit de poids fort est nul, les 7 autres bits contenant ce nombre n.

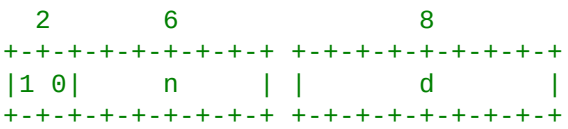
***Chaîne d'octets :***




Chaîne de n octets de données d(1),..., d(n)
Le compte n doit être positif .

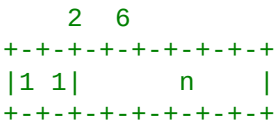
***Octets répliqués :***

Pour compresser une chaîne comportant n répliques de l'octet de données d, les deux octets suivants sont émis :




***Chaîne de bourrage :***

Une chaîne de n octets de bourrage peut être compressée en seulement deux octets, dans lesquels l'octet indiquant la valeur de bourrage change selon le type de représentation de données. Si le type est l'ASCII ou EBCDIC l'octet de bourrage est l'espace <SP> (ASCII code 32, EBCDIC code 64). Si le type est Image ou Local la valeur de bourrage vaut 0.




***Séquence de contrôle :***

Une séquence de contrôle est un octet double, dont le premier est le caractère d'échappement (octet nul) et le deuxième contient les codes de description tels que définis dans le mode Bloc. Le descripteur a la même signification que dans le mode Bloc, et s'applique à la chaîne qui le suit.

Le mode compressé est particulièrement utile pour gagner de la bande passante lors de transferts de gros volumes de données, et ce pour un coût CPU assez faible. Il peut être utilisé de façon très efficace pour transmettre des fichiers de sortie d'impression directement formatés.

### *3.5. RECUPERATION D'ERREUR ET REPRISE DE TRANSMISSION*

Il n'existe pas de mécanisme permettant de détecter des bits perdus ou erronés d'un fichier transféré; ce niveau d'erreur est géré au niveau de TCP. Cependant, une procédure de reprise est prévue pour protéger les utilisateurs de défaillances majeures

des systèmes (incluant le crash d'un hôte, d'un processus FTP, ou d'un protocole réseau sous-jacent).

La procédure de reprise n'est définie que dans les modes de transfert par bloc ou compressé. Elle demande à l'émetteur des données d'envoyer un marquer particulier dans le flux de données incluant des informations de reprise. Ces informations du marqueur n'ont de signification que pour l'émetteur, mais doivent consister en des caractères imprimables au sens du langage utilisé pour le contrôle de la connexion (ASCII ou EBCDIC). Le marqueur peut représenter un comptage de bits, d'enregistrements, ou tout autre information pouvant coder un "point de contrôle". Le récepteur des données, s'il implémente la procédure de reprise, notera la position de ce point au niveau de l'hôte récepteur, et renvoie cette information à l'utilisateur.

Dans le cas d'une faute système, l'utilisateur peut alors enclencher la procédure de reprise en notifiant le point de contrôle. L'exemple suivant illustre l'utilisation de la procédure de reprise.

L'émetteur des données insère un bloc de marquage approprié dans le flux de données en un point donné. Le récepteur des données marque le point de contrôle dans son système de fichiers local et indique les derniers points émis et reçus à l'utilisateur, soit directement, soit en utilisant la réponse de code 110 du protocole de contrôle (suivant qui est l'émetteur). Lors d'une faute système, l'utilisateur ou le contrôleur requiert un nouveau transfert à partir du dernier marqueur en émettant le bloc de reprise avec ce marqueur comme argument. La commande de reprise est transmise via le canal de contrôle et est immédiatement suivi de la commande (telle que RETR, STOR ou LIST) qui était en exécution avant la faute système.

# 4. FONCTIONS DE TRANFERT DE FICHIERS

Le canal de communication entre le USER-PI et le SERVER-PI est établi comme une connexion TCP entre l'utilisateur et le port standard FTP du serveur. L'interpréteur de protocole est responsable de l'émission des commandes FTP et de l'interprétation des réponses; le SERVER-PI interprète les commandes, envoie les réponses, et pilote le DTP pour établir le canal de données et transférer les fichiers. Si le correspondant du processus de transfert (le processus passif) est un USER-DTP, alors celui-ci est lui-même piloté par l'intermédiaire de l'interpréteur de protocole de l'hôte USER-FTP; s'il s'agit d'un second SERVER-DTP, alors son contrôle se fait via son propre PI sur commande du USER-PI. Les réponses FTP sont décrites dans la section suivante. Dans la description des quelques commandes de la section présente, il nous est apparu utile d'être explicite sur les réponses à attendre.

## *4.1. COMMANDES FTP*

Le protocole FTP suit les recommandations du protocole Telnet pour toutes les communications sur le canal de contrôle. Comme le langage choisi pour la communication sous Telnet peut être une option négociée, toutes les références dans les deux prochaines section se font par rapport au "langage Telnet" et le "code de fin de ligne Telnet" correspondant. De façon courante, on considérera qu'il s'agit du NVT-ASCII et de la séquence respective <CRLF>. Aucune autre spécification du protocole Telnet ne sera citée ici.

Les commandes FTP sont des chaînes de caractères "Telnet" terminées par le "code de fin de ligne Telnet". Les codes de commande sont eux-mêmes des caractères alphabétiques suivis du caractère <SP> (Espace) si d'autres paramètres suivent, et Telnet-EOL dans le cas contraire. Les codes et sémantique des commandes sont décrites dans cette section; la syntaxe détaillée est décrite dans la Section traitant des Commandes, les séquences de réponse sont explicitées dans la Section traitant du Séquencement des Commandes et Réponses, et les scénarios illustrant l'usage typique d'une commande sont donnés en Section traitant des Scénarios FTP Typiques.

Les commandes FTP peuvent être divisées en commandes de contrôle d'accès, commandes de paramétrage de transfert, et des commandes de service FTP. Certaines commandes (telles qu'ABOR, STAT, QUIT) peuvent être émises via le canal de contrôle y compris lorsqu'un transfert est en cours. Certains serveur ne pourront simultanément gérer le canal de contrôle et celui de données, auquel cas certaines actions spéciales devront être faites pour attirer l'attention du serveur. La procédure suivante doit être employée dans cet ordre:

1. Le système de l'utilisateur insère un signal "Interrupt Process" Telnet (IP) dans le flux Telnet.

2. Le système utilisateur envoie un signal "Synch" Telnet.

3. Le système utilisateur tente une commande d'avortement (ex., ABOR) dans le flux de commande Telnet.

4. Le SERVER-PI, après réception de "l'IP", inspecte le flux Telnet en attendant EXACTEMENT UNE commande FTP.

(Sur certains serveurs, cette procédure n'est pas indispensable, mais son activation ne produira pas d'effets inattendus).

### 4.1.1. COMMANDES DE CONTROLE D'ACCES

Les commandes qui suivent traitent du paramétrage du contrôle d'accès (les codes numériques de commande sont donnés entre parenthèses).

**USER NAME (USER)** NOM D'UTILISATEUR

Le champ argument est une chaîne Telnet identifiant l'utilisateur. L'identifiant de l'utilisateur est celui qui est requis par le serveur pour permettre l'accès au système de fichiers de l'hôte serveur. Cette commande est normalement la première à être envoyée dès que le canal de contrôle est mis en place (certains serveurs l'imposent). Des informations d'identification supplémentaires telles qu'un mot de passe et/ou un nom de compte utilisateur peuvent être aussi requis par certains serveurs. Les serveurs doivent accepter une nouvelle commande USER à tout moment en vue de changer les droits et privilèges d'accès, ou le compte. Ceci aura l'effet d'annuler toute référence à l'utilisateur, au mot de passe, et au compte précédent en recommençant la séquence d'ouverture de session depuis le début. Tous les paramètres de transfert

restent cependant inchangés et tout transfert de fichier en cours se termine normalement avec les anciens paramètres de session.

**PASSWORD (PASS)** MOT DE PASSE

Le champ argument est une chaîne Telnet indiquant le mot de passe attribué à cet utilisateur. Cette commande doit immédiatement suivre la commande précédente, et, sur certains sites, complète les données d'identification de l'utilisateur pour lui permettre un accès au système de fichiers. Comme le mot de passe est une information dite "sensible", il est préférable de le "masquer" lors de son entrée, voire d'en éviter l'impression en clair à l'écran. Cependant, il apparaît que le serveur n'a aucun moyen de s'opposer à sa divulgation. Il est donc de la responsabilité des USER-FTP d'éviter le stockage explicite du mot de passe et son affichage.

**ACCOUNT (ACCT)** COMPTE UTILISATEUR

Le champ argument est une chaîne Telnet qui spécifie le "compte" de l'utilisateur. Cette commande n'est pas nécessairement couplée à une commande USER, et certains site pourront imposer la spécification d'un compte à l'ouverture de session tandis que d'autre ne le demanderont que pour des accès spécifiques, par exemple pour enregistrer des fichiers. Dans ce dernier cas, il est admis que cette commande puisse arriver à tout moment.

Des codes de réponse existent pour différencier ces cas pour un automate : lorsque l'infirmation de compte est requise à l'ouverture de session, la réponse à une commande PASSword exécutée avec succès est de code 332. Dans l'autre cas où le compte utilisateur n'est pas requis à l'ouverture de session, la réponse donnée à une commande PASSword concluante est de code 230; enfin, si le compte utilisateur est requis à la suite d'une commande exécutée plus loin dans le processus, le serveur répondra par un code 332 ou 532 suivant que la commande précédente est complétée (attente de la commande ACCounT) ou respectivement avortée.

**CHANGE WORKING DIRECTORY (CWD)** CHANGEMENT DE REPERTOIRE

Cette commande permet de changer le répertoire distant de travail (récupération ou téléchargement de fichiers) sans modifier les paramètres en cours de la session. Les paramètres de transfert restent eux aussi inchangés. L'argument est un chemin d'accès valide dans le langage du système de fichier local.

**CHANGE TO PARENT DIRECTORY (CDUP)** ACCES AU REPERTOIRE PERE

Il s'agit d'un cas particulier de la commande CWD, et est définie pour simplifier l'implémentation de programmes transférant des structures entières de répertoires entre des systèmes d'exploitation utilisant des syntaxes différentes pour l'accès au répertoire père. Les codes de réponse attendus sont identiques à ceux attendus pour la commande CWD. Voir l'Appendice II pour plus de détails.

**STRUCTURE MOUNT (SMNT)** MONTAGE DE VOLUME

Cette commande permet de monter un volume sous un système de fichier différent sans changer de contexte pour la session. Les paramètres de transfert sont de même inchangés. L'argument est un chemin d'accès valide du système local.

**REINITIALIZE (REIN)** REINITIALISATION

Cette commande tue une connexion USER, libérant toute les ressources d'entrées/sorties et les informations de session, sauf pour l'opération de transfert en cours qui est achevée normalement. Tous les paramètres sont rétablis dans leurs valeurs par défaut et le canal de contrôle est laissé ouvert. L'état obtenu est identique à l'état dans lequel serait un canal de contrôle juste après son établissement. Une commande USER est en général attendue.

**LOGOUT (QUIT)** FERMETURE DE SESSION

Cette commande termine une session USER et si aucun transfert n'est en cours, ferme le canal de contrôle. Si un fichier est en cours de transfert, la connexion restera ouverte jusqu'à recevoir le code de résultat de l'opération, puis sera fermée par le serveur. Un processus utilisateur qui transfère des fichiers multiples pour des USER distincts sans être obligé de couper puis de rouvrir à chaque fois une nouvelle session, utilisera plutôt une commande REIN.

Une fermeture inopinée du canal de contrôle sera considéré par un serveur comme la succession implicite d'un commande d'avortement (ABOR) suivi d'une fermeture de session (QUIT).

### 4.1.2. COMMANDES DE PARAMETRAGE DU TRANSFERT

Tous les paramètres de transfert ont des valeurs par défaut, et l'usage des commandes de paramétrage du transfert ne sont à utiliser que dans le cas ou des valeurs non standard sont requises pour la connexion. Les valeurs "par défaut" sont usuellement les dernières utilisées, ou, si aucune n'a été spécifiée, la valeur par défaut "standard". Ceci implique que le serveur doit se "rappeler" des valeurs par défaut applicables. Ces commandes peuvent apparaître dans n'importe quel ordre, mais doivent toujours précéder les requêtes de service FTP. Les commandes suivantes spécifient les paramètres de transfert :

**DATA PORT (PORT)** PORT DU CANAL DE DONNEES

L'argument est une spécification de port hôte indiquant le port de données à utiliser pour l'établissement du canal de données. Il existe des valeurs standard pour les ports USER et SERVER, et, dans une situation normale, cette commande et ses réponses associées ne sont pas exploitées. Si cette commande est utilisée, l'argument doit être noté comme la concaténation d'une adresse TCP/IP complètement qualifiée, soit une adresse Internet en 32-bits et une adresse de port TCP en 16-bits . Cette adresse est découpée en champs

de 8-bits dont la valeur est transmise comme un nombre décimal (dans une représentation sous forme chaîne de caractères). Les champs sont séparés par des virgules . Une commande PORT aurait l'allure suivante :

```
PORT h1,h2,h3,h4,p1,p2
```

dans laquelle h1 contient les 8 bits de poids fort de l'adresse Internet de l'hôte spécifié.

## PASSIVE (PASV) — MODE PASSIF

Cette commande demande au SERVER-DTP de se mettre "à l'écoute" d'un port de données (différent du port par défaut) et d'attendre une demande de connexion plutôt que de prendre l'initiative d'en établir une sur réception d'une commande de transfert. La réponse à cette commande précise l'adresse et le port sur lesquels le serveur s'est mis en écoute.

## REPRESENTATION TYPE (TYPE) — TYPE DE REPRESENTATION

L'argument de cette commande spécifie le type de représentation des données utilisée conformément à la Section traitant des Représentation de Données et Stockage. Plusieurs types admettent un second paramètre. Le premier paramètre est exprimé comme un seul et unique caractère Telnet, tout comme le second paramètre Format dans le cas des types ASCII et EBCDIC; le second paramètre dans le cas du type LocalByte est un entier décimal indiquant la taille de l'octet logique. Les paramètres sont séparés par des <SP> (Espace, ASCII code 32).

Les codes suivants sont réservés pour les types :

```
A - ASCII  |    | N - Non-print
           |-><-| T - Telnet format effectors
E - EBCDIC |    | C - Carriage Control (ASA)

I - Image
L <byte size> - taille d'octet logique locale
```

La représentation des données utilisée par défaut est l'ASCII "Non-print". Si le paramètre de Format est modifié, puis le premier argument est à son tour changé, le Format retourne à la valeur "Non-print" par défaut.

## FILE STRUCTURE (STRU) — STRUCTURE DE FICHIER

L'argument est donné sous forme d'un caractère Telnet unique spécifiant la structure de fichier conformément à la Section traitant des Représentations de Données et Stockage.

Les codes suivant sont actuellement réservés pour l'encodage des structures :

```
        F - structure-fichier (pas de structure sous-
jacente)
        R - structure-enregistrement
        P - structure-pages
```

La structure par défaut est la structure-fichier.

**TRANSFER MODE (MODE)** **MODE DE TRANSFERT**

L'argument est donné sous forme d'un caractère Telnet unique spécifiant les modes de transfert de données décrits dans la Section traitant des Modes de Transmission.

Les codes suivants sont réservés pour l'encodage du mode de transmission :

```
        S - flux (stream)
        B - Bloc
        C - Compressé
```

Le mode de transfert par défaut est le mode flux.

### 4.1.3. COMMANDES DE SERVICE FTP

Les commandes de service FTP rassemblent toutes les commandes opérationnelles de transfert ou système qui peuvent être invoquées par l'utilisateur. L'argument d'une commande de service FTP est en général un chemin d'accès. La syntaxe de ce chemin doit se conformer aux conventions adoptées par le site serveur (avec une valeur par défaut applicable), et aux conventions de langage adoptée par le canal de contrôle. La valeur par défaut conseillée est soit la dernière combinaison d'unité logique, chemin d'accès et nom de fichier, soit un chemin complet défini comme défaut par l'utilisateur. Les commandes peuvent être invoquées dans n'importe quel ordre excepté pour le couple "rename from", "rename to" qui doit être exécuté dans cette ordre et subséquemment, et le cas de la commande "restart" qui doit être suivie de la dernière commande avortée (ex., STOR ou RETR). Les données, lorsqu'elles sont émises en réponse à une commande de service FTP, devront toujours l'être via le canal de données, sauf pour certaines réponses à caractère informatif. Les commandes suivantes dont partie de la classe "commandes de service FTP" :

**RETRIEVE (RETR)** TRANSMISSION

Cette commande provoque la transmission par le SERVER-DTP d'une copie du fichier spécifié par son chemin d'accès complet, à destination du

SERVER- ou USER-DTP à l'autre extrémité du canal de données. Le statut et le contenu du fichier côté émetteur doit rester inchangé.

**STORE (STOR)** ENREGISTREMENT

Cette commande provoque l'acceptation par le SERVER-DTP des données transférées via le canal de données, lesquelles seront enregistrées dans un fichier sur le serveur récepteur. Si le fichier entièrement spécifié existe sur le serveur avant la transmission, alors son contenu sera remplacé par le contenu transmis. Dans l'alternative, un nouveau fichier est créé.

**STORE UNIQUE (STOU)** ENREGISTREMENT UNIQUE

Cette commande provoque le même comportement que la commande STOR excepté le fait que le fichier doit être créé dans le répertoire courant sous un nom unique. La réponse de code 250 (Transfer Started) doit inclure le nom de fichier généré par le site récepteur.

**APPEND (with create) (APPE)**

Cette commande provoque l'acceptation par le SERVER-DTP des données transmises sur le canal de données, lesquelles seront enregistrées dans un fichier sur le site de réception. La différence avec la commande STOR réside dans le fait que si le fichier spécifié existe déjà sur le site de réception, les données transmises viennent s'ajouter au fichier existant.

**ALLOCATE (ALLO)** ALLOCATION

Cette commande peut être nécessaire sur certains serveurs pour réserver un espace de stockage suffisant pour permettre le stockage des données à transférer. L'argument est un entier donnant la taille en octets à réserver (la taille est relative à l'octet logique). Pour des fichiers transférés en mode enregistrement ou par pages, un nombre maximal d'enregistrement ou une taille maximale de page (comptée en octets logiques) peut être nécessaire; ces valeurs sont indiquées par l'usage d'un deuxième paramètre entier décimal. Ce second argument est optionnel, et doit être séparé du premier, lorsqu'utilisé, par les trois caractères Telnet <SP> R <SP>. Cette commande doit être usuellement suivie d'une commande STORe ou APPEnd. La commande ALLO doit être traitée comme une commande NOOP (no operation) par tous les serveurs ne nécessitant pas une prédéclaration de la taille de fichiers à enregistrer, ceux qui nécessitent seulement une mention de la taille maximale d'enregistrement ou de taille maximale de page peuvent accepter une absence de valeur pour le premier paramètre, ou ignoreront la valeur si spécifiée.

**RESTART (REST)** REPRISE

Le champ argument contient une expression du marqueur de contrôle à partir duquel le transfert doit être repris. Cette commande ne provoque pas explicitement de transfert de données, mais déplace simplement le point de

lecture du fichier interrompu jusqu'au point de contrôle spécifié. Cette commande sera immédiatement suivie de la commande de service FTP nécessaire à relancer le processus de transfert.

**RENAME FROM (RNFR)** RENOMMER...

Cette commande indique l'ancien chemin d'accès complet du fichier qui doit être renommé. Cette commande doit être immédiatement suivie d'une commande "rename to" spécifiant le nouveau nom du fichier en question.

**RENAME TO (RNTO)** RENOMMER VERS...

Cette commande indique le nouveau nom du fichier spécifié dans le commande "rename from" précédente. L'usage subséquent de ces deux commandes provoque le changement du nom du fichier sur le système distant.

**ABORT (ABOR)** AVORTEMENT

Cette commande provoque l'interruption immédiate de la dernière commande de service FTP et tout transfert de données associé. Cette commande peut demander une "action spéciale", comme il est discuté dans le Section traitant des Commandes FTP, pour en forcer la reconnaissance asynchrone par le serveur. Aucune action n'est a effectuer si la commande précédent a été achevée (y compris un transfert de données). Le canal de contrôle ne doit pas être coupé par le serveur, mais le canal de données doit être fermé.

Le serveur doit prendre en compte deux situations sur réception de cette commande : (1) toute commande de service FTP est achevée, ou (2) une commande de service FTP est en cours. Dans le premier cas, le serveur ferme le canal de données (s'il est encore ouvert) et répond par un code 226, indiquant que la commande d'avortement a été correctement traitée.

Dans le second cas, le serveur interrompt le service FTP en cours, coupe le canal de données, et renvoie un code 426 pour indiquer que la dernière commande s'est achevée anormalement. Le serveur envoie à la suite un code 226, indiquant que la commande d'avortement elle-même s'est bien déroulée.

**DELETE (DELE)** SUPPRESSION

Cette commande provoque la suppression sur le site serveur du fichier précisé par le chemin d'accès complet. Si une étape supplémentaire de protection est nécessaire (telle qu'une confirmation éventuelle du type "Supprimer réellement ce fichier?"), elle doit être fournie par le processus USER-FTP.

**REMOVE DIRECTORY (RMD)** SUPPRESSION DE REPERTOIRE

Cette commande provoque la suppression du chemin d'accès spécifié au titre de répertoire (si le chemin est absolu) ou de sous répertoire du répertoire courant (si le chemin est relatif). Voir l'appendice II.

**MAKE DIRECTORY (MKD)** CREATION DE REPERTOIRE

Cette commande provoque la création d'un répertoire (si le chemin est absolu) ou d'un sous répertoire du répertoire courant (si le chemin est relatif) selon le chemin spécifié. Voir l'appendice II.

**PRINT WORKING DIRECTORY (PWD)** IMPRESSION DU REPERTOIRE COURANT

Cette commande renvoie le nom du répertoire courant dans la réponse. Voir Appendice II.

**LIST (LIST)** CATALOGUE DU REPERTOIRE COURANT

Cette commande provoque l'émission par le serveur d'une liste de fichiers au DTP passif. Si le chemin mentionné spécifie un répertoire ou tout autre groupe de fichiers, le serveur répondra par une liste des fichiers dans ce répertoire ou ce groupe. Si le chemin spécifie un fichier normal, alors les informations système relatives à ce fichier seront renvoyées. Une absence d'argument indique par défaut le répertoire courant. La réponse est transférée via le canal de données pour les types ASCII ou EBCDIC. (l'utilisateur doit s'assurer que le type est effectivement ASCII ou EBCDIC). Comme les informations relatives à un fichier peuvent varier grandement en forme et présentation entre divers systèmes, celles-ci seront généralement peu exploitable par un automate. Elles sont cependant fort utiles pour un utilisateur humain.

**NAME LIST (NLST)** CATALOGUE COURT

Cette commande provoque l'envoi par le serveur d'un catalogue succinct d'un de ses répertoires vers l'utilisateur. Le chemin spécifié doit décrire un répertoire valide ou tout autre descripteur d'un ensemble de fichiers; un argument omis désigne le répertoire courant. Le serveur répondra par une liste de noms de fichiers à l'exclusion de toute autre information. Les données sont transférées en ASCII ou EBCDIC sur le canal de données sous forme d'une suite de noms de chemins d'accès valides séparés par des <CRLF> ou <NL>. (Encore une fois, l'utilisateur doit s'assurer que le paramètre TYPE est correct). Cette commande a été implémentée pour permettre à des processus automatiques de pouvoir récupérer cette liste pour traitement ultérieur. Un cas typique est l'implémentation d'une fonction de téléchargement de fichiers multiples.

**SITE PARAMETERS (SITE)** PARAMETRES DE TAILLE

Cette commande est utilisée par le serveur pour proposer des services spécifiques à ce système qui sont indispensables pour le transfert de fichiers mais insuffisamment universels pour justifier l'attribution d'une commande dans le protocole. La nature de ces services, et leur syntaxe devra être fournie par chaque service les utilisant, en réponse d'une commande HELP SITE.

**SYSTEM (SYST)** SYSTEME

Cette commande permet de connaître le type de système d'exploitation sur le serveur. La réponse devra mentionner dans son premier "mot" l'un des systèmes mentionnés dans le document Assigned Numbers [4] en cours de validité.

**STATUS (STAT)** STATUT

Cette commande provoque l'envoi d'un message d'état (statut) de réponse sur le canal de contrôle. Cette commande peut être utilisée en cours de transfert (avec les signaux IP et Synch de Telnet – voir la Section traitant des commandes FTP) auquel cas le serveur doit répondre avec l'état de la transaction en cours, ou bine elle peut être envoyée entre deux transferts. Dans ce dernier cas, la commande devra être utilisée avec un argument. Si cet argument est un chemin d'accès, la commande résultante équivaut à une commande "list" à l'exception près que la réponse sera transmise par le canal de connexion au lieu du canal de données. Si un chemin partiel est donné, Le serveur répondra par une liste de noms de fichiers ou d'attributs associés à cette spécification. Si aucun argument n'est donné, le serveur renverra une information générale concernant le processus serveur FTP. Ceci pourra inclure l'ensemble des paramètres de connexion actuellement utilisé ainsi que l'état de toutes les connexions.

**HELP (HELP)** AIDE

Cette commande provoque l'envoi d'une information d'aide concernant l'implémentation du serveur lui-même, via la connexion de contrôle. Cette commande peut prendre un argument (ex., n'importe quel nom de commande) et renvoie des informations encore plus précises. La réponse sera de type 211 ou 214. Il est suggéré que la commande HELP soit permise y compris avant qu'une commande USER d'ouverture de session n'ait été exécutée. Le serveur pourra utiliser cette commande pour donner des informations sur des paramètres dépendants du système, ex., en réponse à la requête "HELP SITE".

**NOOP (NOOP)** PAS D'ACTION

Cette commande n'affecte aucun paramètre ni n'interagit avec aucune des commandes précédemment lancées. Elle provoque aucune autre action qu'une simple réponse "OK" de la part du serveur.

### *4.2. REPONSES FTP*

Les réponses à des commandes FTP sont destinées à assurer une certaine synchronisation des actions impliquées dans un processus de transfert de fichiers, et garantir que le processus utilisateur puisse toujours connaître l'état du serveur. Chaque commande suscite au moins une réponse, mais plusieurs réponses peuvent être

données; dans ce dernier cas, les multiples réponses devront être aisément différentiables. De plus, certaines commandes peuvent être émises groupées en séquence, comme USER, PASS et ACCT, ou RNFR et RNTO. Les réponses témoignent de l'existence d'états intermédiaires si toutes les commandes passées sont exécutées avec succès. L'échec d'une seule étape nécessitera de recommencer toute la procédure.

Les détails d'une séquence de commandes-réponses sont explicitées dans l'ensemble de diagrammes ci-après.

Une réponse FTP répond consiste en un nombre à trois chiffres (transmis sous forme de trois caractères alphanumériques) suivis d'un texte. Le code numérique est à destination d'automates pour renseigner des dispositions à prendre et de l'état suivant de celui-ci; le texte est plutôt destiné à l'utilisateur humain. Les trois digits du code sont sensés contenir suffisamment d'information pour que le processus utilisateur (USER-PI) n'ait pas nécessité d'examiner la partie texte de la réponse, laquelle peut être soit éliminée, soit transférée à l'interface utilisateur, selon la nécessité. En particulier, le texte émis peut varier de serveur à serveur, et un automate pourrait donc avoir des difficultés à analyser tous les messages possibles.

Une réponse est définie comme contenant le code à 3-digits, suivi d'un Espace <SP>, suivie par une ligne de texte (lorsqu'une longueur maximale de réponse a été définie auparavant), et terminée par le code de fin-de-ligne Telnet. Il y aura des cas cependant, ou le texte sera plus long qu'une simple ligne. Dans ce cas, le texte entier aurait pu être mis entre crochets de sorte que le processus utilisateur puisse savoir quand s'arrête la lecture du texte (c-à-d. arrête l'analyse de l'entrée du canal de contrôle) pour passer à d'autres tâches. Ceci implique l'utilisation d'un format particulier sur la première ligne pour indiquer que d'autres lignes suivent, et un autre format particulier sur la dernière. Au moins une de ces lignes doit présenter le code de réponse. Pour satisfaire tous les avis sur le problème, il a été décidé que le code serait identique sur la première et la dernière ligne.

Ainsi, le format d'une réponse multilignes est tel que la première ligne débute par le code exact de la réponse, suivi d'un tiret "-" (Hyphénation ou "moins"), suivi du texte de la première ligne. La dernière ligne commencera par le même code, suivi immédiatement d'un Espace <SP>, éventuellement du texte, terminé par le code de fin-de-ligne Telnet.

Par exemple:

```
123-First line
Second line
 234 A line beginning with numbers
123 The last line
```

Le processus utilisateur n'a plus qu'à chercher la deuxième occurrence du code de réponse suivie de l'Espace <SP> en début de ligne, et ignorer les lignes intermédiaires. Si une ligne intermédiaire commence par un nombre de 3-digits, le serveur ajoutera un espace en tête de ligne pour éviter toute confusion.

Ce schéma permet à des routines système standard d'être employées pour générer la réponse (ex. pour la réponse à la commande STAT), avec un marquage supplémentaire "artificiel" en tête de la première et la dernière ligne. Au cas (rare) ou ces routines seraient susceptibles de générer une ligne commençant par 3 digits suivi d'un espace, un caractère neutre (ex. Espace) sera rajouté en tête de chaque ligne.

Ce schéma permet d'éviter la mise entre crochets de la réponse.

Les trois digits de la réponse ont chacun une signification particulière. Ceci permet d'implémenter des traitements à réponse du plus simple au plus complexe dans l'USER-PI. Le premier digit indique si la commande se termine en succès, échec, ou est incomplète. (Rapport au diagramme d'état), un interpréteur de protocole simpliste pourra déterminer une stratégie d'action à lancer (telles que se retirer, tenter de nouveau, etc.) en se bornant à examiner ce digit. Un processus utilisateur désireux de savoir de quelle nature est l'erreur, (ex. erreur du système de fichiers, erreur de syntaxe dans la commande) pourra examiner le second digit, le troisième étant réservé au degré le plus fin de signalisation (ex., une commande RNTO sans commande RNFR antérieure).

Le premier digit peut prendre 5 valeurs différentes :

**1yz Réponse positive préliminaire**

L'action demandée a été correctement reconnue et lancée; on devra attendre une autre réponse pour pouvoir demander l'exécution d'une nouvelle commande. (Un processus utilisateur émettant une nouvelle commande avant conclusion de la première obtiendrait une réponse d'erreur du type "violation de protocole"; certains processus serveur FTP peuvent empiler les réponses entrantes sans émettre ce type d'avertissement). Ce type de réponse est utilisé pour avertir l'utilisateur que sa commande a été bien reconnue et qu'il peut alors surveiller son canal de données, notamment dans le cas d'applications dans lesquelles la surveillance simultanée des deux canaux "contrôle" et "données" n'est pas pratique. Un serveur FTP devra au moins émettre une commande de classe 1yz par commande reçue.

**2yz Réponse positive définitive**

L'action demandée s'est complètement déroulée avec succès. Une nouvelle commande peut être reçue par le serveur.

**3yz Réponse positive intermédiaire**

La commande a été acceptée, mais le serveur a mis celle-ci en sommeil, dans l'attente d'informations supplémentaires. L'utilisateur devra alors émettre une autre commande avec les informations demandées. Cette réponse est utilisée dans les groupements de commandes en séquence.

**4yz Réponse négative transitoire**

La commande a été refusée, et l'action n'a pas été exécutée, mais la condition d'erreur invoquée est de nature temporaire, impliquant que la même

commande peut être tentée à nouveau. Dans le cas d'une séquence de commandes groupées, l'utilisateur reprendra toute la séquence depuis son début. Le contexte du terme "transitoire" reste cependant difficile à expliciter, en particulier lorsque deux sites distincts (SERVER- et processus USER) doivent s'accorder sur son interprétation. Chaque réponse de la classe 4yz peut correspondre à un contexte de durée différent, mais le but de cette classe est de signaler au processus utilisateur la possibilité de tenter l'opération encore une fois. Une règle d'implémentation pour savoir si une réponse doit entrer ou doit être fournie dans la classe 4yz ou 5yz (Négative définitive) est la suivante : une réponse sera de classe 4yz si la commande peut être répétée avec une chance de succès, A L'IDENTIQUE, et sans aucune modification des paramètres USER ou SERVER (c-à-d., la commande est écrite strictement comme la première; l'utilisateur ne change pas ses droits d'accès, ne change pas de compte ni de session; le serveur ne change pas d'implémentation).

**5yz Réponse négative définitive**

La commande a été refusée, et l'action n'a pas été exécutée. Le serveur notifie par là au processus utilisateur qu'il sera vain de retenter la même commande (dans la même séquence). Certaines conditions d'erreur "permanentes" pourront toutefois être corrigées, et la commande pourra être relancée par une action explicite de l'utilisateur humain, soit après correction de la commande, soit après changement de ses droits, soit après intervention de l'opérateur du serveur.

Le second digit donne une indication sur la nature de la réponse :

**x0z Syntaxe**

Ces réponses se réfèrent à des erreurs de syntaxe, des commandes correctes en termes de syntaxe, mais ne se référant à aucune fonction connue ou implémentée.

**x1z Information**

Indiquent une réponse à des demandes d'information, comme les commandes d'états ou d'aide.

**x2z Connexions**

Réponses se réfèrent à une problématique de connexion sur les canaux "contrôle" ou "données".

**x3z Identification et authentification**

Réponses du processus d'accès au système de fichiers.

**x4z Non encore spécifiée.**

**x5z Système de fichiers**

Ces réponses se réfèrent à l'état du système de fichiers serveur lorsque des commandes de ce système sont invoquées.

Le troisième digit permet de qualifier encore plus finement les réponses dans chacune des catégories données par le deuxième digit. La liste des réponses donnée ci-après le montre. Notez que le contenu informationnel du texte ci-dessous est une "recommandation", et est nature à interprétation en fonction du serveur qui l'émet. Les codes de réponse, par opposition, doivent suivre à la lettre les spécifications indiquées; c'est-à-dire que les implémentations des serveurs ne doivent jamais inventer de nouveaux codes, même si les situations dans lesquelles ils peuvent être sont légèrement différentes que celles définies; elles devront impérativement choisir le code correspondant à la situation la plus proche.

Une commande telle que TYPE ou ALLO dont l'exécution complète n'est pas de nature à apporter une information utile pour le processus utilisateur provoqueront le retour d'une réponse de code 200. Lorsque la commande en question n'est pas implémentée par un processus SERVER-FTP particulier (cette fonction n'a pas de signification dans ce contexte particulier de serveur, par exemple, la commande ALLO sur un site TOPS20), ce dernier devra de préférence répondre par un code positif de sorte que l'utilisateur puisse poursuivre sa procédure. Une réponse de code 202 sera utilisé dans ce cas, associé par exemple au texte suivant: "Allocation non nécessaire." Si, par contre, la commande est générale, mais non implémentée par le site serveur, un code 502 sera répondu. Une version affinée de cette réponse est le code 504 qui précise que cette commande est implémentée, mais l'un au moins des paramètres associés ne l'est pas.

4.2.1 CODES DE REPONSE PAR GROUPES DE FONCTIONS

200 Commande conclue.
500 Erreur de syntaxe, commande non reconnue. Inclut le cas d'une ligne de commande trop longue.
501 Erreur de syntaxe dans le paramètres ou arguments.
202 Commande non implémentée, ou superflue sur ce site.
502 Commande non implémentée.
503 Mauvaise séquence de commandes.
504 Commande non implémentée pour ce paramètre.

110 Réponse à marqueur de reprise. Dans ce cas, le texte doit être exact et n'est pas "adaptable" par des implémentations "locales"; il DOIT indiquer: MARK yyyy = mmmm où yyyy est le marqueur du flux de données USER-DTP, et mmmm le marqueur équivalent côté serveur (noter l'espace indispensable entre les marqueurs et le "=").
211 Statut système, ou réponse d'aide système.
212 Statut de répertoire.
213 Statut de fichier.

214 Message d'aide.
Sur la manière d'utiliser le serveur ou la signification d'une commande non standard. Cette réponse n'est destinée qu'à un utilisateur humain.

215 NOM de type de système.
Le nom de type de système est un nom officiel standard défini dans la RFC "Assigned Numbers".

120 Service disponible dans nnn minutes.

220 Service disponible pour nouvel utilisateur.

221 Canal de contrôle fermé par le service. Cas archivé si nécessaire.

421 Service non disponible, canal de contrôle fermé. Répondu à toute commande lorsque la fermeture imminente du service est prévue.

125 Canal de données déjà ouvert; début de transfert.

225 Canal de données ouvert; pas de transfert en cours.

425 Erreur d'ouverture du canal de données.

226 Fermeture du canal de données. Service terminé (par exemple, transfert de fichier ou avortement).

426 Connexion fermée, transfert interrompu.

227 Passage en mode passif (h1,h2,h3,h4,p1,p2).

230 Session ouverte.

530 Session non ouverte.

331 Nom d'utilisateur reçu, mot de passe demandé.

332 Compte utilisateur demandé.

532 Compte utilisateur demandé pour enregistrement de fichiers.

150 Statut de fichier vérifié; ouverture de canal de données en cours.

250 Service fichier terminé.

257 "CHEMIN" créé.

350 Service fichier en attente d'information.

450 Service fichier non traité. Fichier non disponible (ex., fichier verrouillé par un autre utilisateur).

550 Service fichier non traité. Fichier non accessible (ex., fichier non trouvé, accès refusé).

451 Service interrompu. Erreur locale de traitement.

551 Service interrompu. Type de page inconnu.

452 Service interrompu. Espace insuffisant.

552 Service fichier interrompu. Quota dépassé (pour le répertoire ou compte courant).

553 Service interrompu. Nom de fichier erroné.

### 4.2.2 CODES REPONSE PAR ORDRE NUMERIQUE

110 Réponse à marqueur de reprise. Dans ce cas, le texte doit être exact et n'est pas "adaptable" par des implémentations "locales"; il DOIT indiquer: MARK yyyy = mmmm où yyyy est le marqueur du flux de données USER-DTP, et mmmm le marqueur équivalent côté serveur (noter l'espace indispensable entre les marqueurs et le "=").

120 Service disponible dans nnn minutes.

125 Canal de données déjà ouvert; début de transfert.

150 Statut de fichier vérifié; ouverture de canal de données en cours.

200 Commande conclue.
202 Commande non implémentée, ou superflue sur ce site.
211 Statut système, ou réponse d'aide système.
212 Statut de répertoire.
213 Statut de fichier.
214 Message d'aide.
Sur la manière d'utiliser le serveur ou la signification d'une commande non standard. Cette réponse n'est destinée qu'à un utilisateur humain.
215 NOM de type de système.
Le nom de type de système est un nom officiel standard défini dans la RFC "Assigned Numbers".
220 Service disponible pour nouvel utilisateur.
221 Canal de contrôle fermé par le service. Cas archivé si nécessaire.
225 Canal de données ouvert; pas de transfert en cours.
226 Fermeture du canal de données. Service terminé (par exemple, transfert de fichier ou avortement).
227 Passage en mode passif (h1,h2,h3,h4,p1,p2).
230 Session ouverte.
250 Service fichier terminé.
257 "CHEMIN" créé.

331 Nom d'utilisateur reçu, mot de passe demandé.
332 Compte utilisateur demandé.
350 Service fichier en attente d'information.

421 Service non disponible, canal de contrôle fermé. Répondu à toute commande lorsque la fermeture imminente du service est prévue.
425 Erreur d'ouverture du canal de données.
426 Connexion fermée, transfert interrompu.
450 Service fichier non traité. Fichier non disponible (ex., fichier verrouillé par un autre utilisateur).
451 Service interrompu. Erreur locale de traitement.
452 Service interrompu. Espace insuffisant.

500 Erreur de syntaxe, commande non reconnue. Inclut le cas d'une ligne de commande trop longue.
501 Erreur de syntaxe dans le paramètres ou arguments.
502 Commande non implémentée.
503 Mauvaise séquence de commandes.
504 Commande non implémentée pour ce paramètre.
530 Session non ouverte.
532 Compte utilisateur demandé pour enregistrement de fichiers.
550 Service fichier non traité. Fichier non accessible (ex., fichier non trouvé, accès refusé).
551 Service interrompu. Type de page inconnu.
552 Service fichier interrompu. Quota dépassé (pour le répertoire ou compte courant).

553 Service interrompu. Nom de fichier erroné.

# 5. SPECIFICATIONS DECLARATIVES

## *5.1. IMPLEMENTATION MINIMALE*

Afin de rendre FTP exploitable sans multiplication de messages d'erreur inutiles, une implémentation minimale à assurer par tous serveurs est définie ci-après :

```
TYPE - ASCII Non-print
MODE - Flux
STRUCTURE - Fichier, Enregistrement
COMMANDES - USER, QUIT, PORT,
     TYPE, MODE, STRU, pour les valeurs par défaut
     RETR, STOR,
     NOOP.
```

Les paramètres par défaut pour le transfert de données sont :

```
TYPE - ASCII Non-print
MODE - Stream
STRU - File
```

Tous les hôtes doivent accepter les paramètres ci-dessus comme paramètres par défaut standards.

## *5.2. CONNEXIONS*

L'interpréteur de protocole serveur "écoute" sur le Port L. L'utilisateur, ou son interpréteur de protocole doit prendre l'initiative de l'établissement de la connexion bidirectionnelle du canal de contrôle. Les processus SERVER- et USER- doivent suivre les conventions du protocole Telnet spécifiées dans le ARPA-Internet Protocol Handbook [1]. Les serveurs n'ont aucune obligation de fournir un moyen d'éditer la ligne de commande. Cette édition, si nécessaire, peut être déléguée au processus utilisateur. La connexion de contrôle pourra être coupée par le serveur sur demande de l'utilisateur, après conclusion de toutes les transmissions de commandes et réponses associées.

Le USER-DTP doit "écouter" le Port de données spécifié; celui-ci peut être le port standard (U) ou le port donnée par une commande PORT. Le serveur peut lui-même établir la connexion du canal de données entre son port de données standard (L-1) et le port utilisateur spécifié. Le port utilisé, et le sens de transfert seront déterminés par la nature de la commande de service FTP à exécuter.

Notez que toutes les implémentations FTP doivent pouvoir piloter des transferts sur le port de données par défaut, alors que seuls les USER-PI peuvent provoquer l'usage des ports non standard.

Lorsque des données doivent être transmises entre deux serveurs, A et B (voir Figure 2), le USER-PI de C établit un canal de contrôle avec chacun des SERVER-PI

de A et respectivement de B. L'un des serveurs, disons A, reçoit une commande PASV lui indiquant "d'écouter" son port de données plutôt que tenter une connexion lorsqu'il reçoit un ordre de transfert. Lorsque le USER-PI reçoit l'acquittement de cette commande PASV, qui inclut l'identité et le port du serveur "écouté", le USER-PI envoie la référence au port A au serveur B par une commande PORT; une réponse est attendue. Le USER-PI émet alors les commandes de service FTP correspondantes à A et à B. Le serveur B établit la connexion de données et le transfert commence. La séquence de commandes est montrée ci-dessous (les messages sont explicités selon un synchronisme vertical, la disposition horizontale restant asynchrone) :

```
 USER-PI - Serveur A          USER-PI - Serveur B
 -------------------          -------------------

C->A : Connect                      C->B : Connect
C->A : PASV
A->C : 227 Entering Passive Mode. A1,A2,A3,A4,a1,a2
                                    C->B : PORT A1,A2,A3,A4,a1,a2
                                    B->C : 200 Okay
C->A : STOR                         C->B : RETR
          B->A : Connect to HOST-A, PORT-a
```

Figure 3

La connexion de données peut être fermée par le serveur sous les conditions décrites à la Section traitant de l'Etablissement de la Connexion de Données. Si la fermeture du canal de données survient après un transfert pour lequel la fermeture du canal ne correspond pas à une nécessité de signaler une fin-de-fichier (EOF), le serveur peut en prendre l'initiative immédiate. Il n'est pas permis d'accepter une nouvelle commande de transfert dans ce cas car le processus utilisateur ne pourra tester l'état de la connexion de données pour se mettre en écoute sur celle-ci (il l'a déjà fait une fois, et n'a pas à priori de moyen de savoir que la connexion est refermée, par contre, un utilisateur doit nécessairement pouvoir se mettre en écoute d'un canal de données "fermé" AVANT d'envoyer une commande de transfert). Pour éviter un blocage en ce point, le serveur renvoie une réponse (226) après avoir fermé le canal de données (ou, si le canal est laissé ouvert, une réponse "transfert terminé" (250)). Le USER-PI devra attendre une de ces deux réponses avant de relancer une commande de transfert

Chaque fois qu'un utilisateur ou un serveur s'aperçoit que le canal de données a été fermé à l'autre bout, il devra le plus rapidement possible vider les tampons associés à ce canal, et refermer son propre côté de ce dernier.

### *5.3. COMMANDES*

Les commandes sont des chaînes de caractères conformes au protocole Telnet transmises sur des canaux de contrôle tel que le décrit la Section traitant des Commandes FTP. Les fonctions et sémantique des commandes sont décrites dans la Section traitant des Commandes de Contrôle d'Accès, Commandes de Paramétrage de Transfert, Commandes de Service FTP, et Commandes Diverses. La syntaxe des commandes est précisée ci-après.

Toute commande commence par un code de commande suivi d'un champ d'argument. Le code de commande est composé d'au plus quatre caractères alphabétiques. Il ne doit pas être tenu compte de la casse dans un code de commande FTP. Ainsi, toutes les combinaisons suivantes se réfèrent à la commande de téléchargement :

```
RETR  Retr  retr  ReTr  rETr
```

Ceci s'applique à tous les symboles utilisés pour donner les codes de valeurs des arguments, comme "A" ou "a" pour le TYPE ASCII. Le code de commande et les arguments doivent être séparés l'un l'autre par un Espace.

Le champ d'argument est une chaîne de longueur variable terminée par la séquence <CRLF> en représentation NVT-ASCII, ou la séquence de fin-de-ligne correspondante si un autre langage a été négocié pour la transmission. Il doit être noté ici que le serveur doit attendre la réception de cette séquence avant d'exécuter une quelconque action.

La syntaxe est explicitée ci-dessous en NVT-ASCII. Tous les caractères du champ d'arguments sont des caractères ASCII incluant toute représentation décimale d'entiers en ASCII. Les crochets indiquent un argument optionnel. Si cette option n'est pas explicite, la valeur par défaut est considérée.

### 5.3.1. COMMANDES FTP

Les commandes FTP sont spécifiées ci-dessous :

```
USER <SP> <nom d'utilisateur> <CRLF>
PASS <SP> <mot de passe> <CRLF>
ACCT <SP> <compte utilisateur> <CRLF>
CWD <SP> <chemin d'accès> <CRLF>
CDUP <CRLF>
SMNT <SP> <chemin d'accès> <CRLF>
QUIT <CRLF>
REIN <CRLF>
PORT <SP> <port> <CRLF>
PASV <CRLF>
TYPE <SP> <code type> <CRLF>
STRU <SP> <code structure> <CRLF>
MODE <SP> <code mode> <CRLF>
RETR <SP> <chemin d'accès> <CRLF>
STOR <SP> <chemin d'accès> <CRLF>
STOU <CRLF>
APPE <SP> <chemin d'accès> <CRLF>
ALLO <SP> <entier décimal>
  [<SP> R <SP> <entier décimal>] <CRLF>
REST <SP> <marqueur> <CRLF>
RNFR <SP> <chemin d'accès> <CRLF>
RNTO <SP> <chemin d'accès> <CRLF>
ABOR <CRLF>
DELE <SP> <chemin d'accès> <CRLF>
RMD <SP> <chemin d'accès> <CRLF>
```

```
MKD <SP> <chemin d'accès> <CRLF>
PWD <CRLF>
LIST [<SP> <chemin d'accès>] <CRLF>
NLST [<SP> <chemin d'accès>] <CRLF>
SITE <SP> <chaîne> <CRLF>
SYST <CRLF>
STAT [<SP> <chemin d'accès>] <CRLF>
HELP [<SP> <chaîne>] <CRLF>
NOOP <CRLF>
```

5.3.2. ARGUMENTS DE COMMANDES FTP

Syntaxe des champs d'arguments ci-avant (notation BNF si applicable) :

```
<nom d'utilisateur> ::= <chaîne>
<mot de passe> ::= <chaîne>
<compte utilisateur> ::= <chaîne>
<chaîne> ::= <char> | <char><chaîne>
<char> ::= tout caractère ASCII de code 0 à 128 sauf
<CR> et <LF>
<marqueur> ::= <pr-chaîne>
<pr-chaîne> ::= <pr-char> | <pr-char><pr-chaîne>
<pr-char> ::= caractères imprimables, codes ASCII de
33 à 126
<taille d'octet> ::= <nombre>
<port> ::= <adresse hôte>,<numéro port>
<adresse hôte> ::= <nombre>,<nombre>,<nombre>,<nombre>
<numéro port> ::= <nombre>,<nombre>
<nombre> ::= tout entier décimal entre 1 et 255
<code format> ::= N | T | C
<type-code> ::= A [<sp> <code format>]
       | E [<sp> <code format>]
       | I
       | L <sp> <taille d'octet>
<code structure> ::= F | R | P
<code mode> ::= S | B | C
<chemin d'accès> ::= <chaîne>
<entier décimal> ::= tout entier décimal codé en ASCII
```

### *5.4. SEQUENCEMENT DES COMMANDES ET REPONSES*

La communication entre l'utilisateur et le serveur est prévue pour autoriser un dialogue bidirectionnel. A ce titre, l'utilisateur émet une commande FTP à laquelle le serveur répond par une première réponse rapide provisoire. L'utilisateur devra attendre cette réponse, positive ou négative, avant de pouvoir émettre une autre commande.

Certaines commandes nécessitent l'attente d'une deuxième réponse. Ces réponses peuvent, par exemple, donner un état d'avancement ou de conclusion d'un transfert de

fichier ou signaler la fermeture du canal de données. Ce sont des réponses secondaires à des commandes de transfert de fichier.

Un groupe important de réponses est celui qui contient les réponses d'information suite à la conclusion d'une connexion. En des circonstances normales, un serveur émettra une réponse de code 220, "attente d'entrée", lorsque la connexion est établie. L'utilisateur devra attendre cette réponse d'état avant d'émettre toute commande. Si les serveur n'est pas immédiatement en état de recevoir des commandes, il émettra une réponse de code 120 "service retardé" immédiatement après l'établissement de la connexion, puis une réponse de code 220 dès qu'il est en état de recevoir. L'utilisateur est averti de ce temps d'attente, et pourra choisir de maintenir la connexion.

### *Réponses spontanées*

Parfois, "le système" a spontanément un message à émettre vers l'utilisateur (en général à tous les utilisateurs connectés).Par exemple, "Arrêt du système dans 15 minutes". FTP ne propose pas de mécanisme particulier pour émettre spontanément un message du serveur vers l'utilisateur. Il est recommandé d'empiler cette réponse au niveau du SERVER-PI et de délivrer cette information lors de l'émission d'une réponse classique ultérieure (par exemple en constituant une réponse multilignes).

La table ci-après liste les réponses positives et négatives pour chaque commande. Il est demandé un respect strict des codes de réponse dans les situations indiquées; le texte émis par le serveur est libre, bien que son sens général et l'indication sur l'action à exécuter dusse rester explicite, dans le contexte, et univoque.

### *Séquences de Commandes Réponses*

Dans cette section sont présentées les séquences usuelles de commandes-réponses. Chaque commande est listée avec toutes ses réponses possibles; les groupes de commandes sont listés ensembles. Les réponses provisoires sont listées en premier, puis les réponses définitives positives et négatives, enfin, les réponses intermédiaires avec les reste des réponses suivantes de la séquence. Ces listes sont à la base des diagrammes d'états, présentés plus en avant.

Etablissement de connexion

```
      120
      220
      220
      421
```

Ouverture de session

```
      USER
       230
       530
       500, 501, 421
       331, 332
      PASS
       230
       202
       530
       500, 501, 503, 421
```

```
 332
ACCT
 230
 202
 530
 500, 501, 503, 421
CWD
 250
 500, 501, 502, 421, 530, 550
CDUP
 200
 500, 501, 502, 421, 530, 550
SMNT
 202, 250
 500, 501, 502, 421, 530, 550
```

Fermeture de session

```
REIN
 120
 220
 220
 421
 500, 502
QUIT
 221
 500
```

Paramètres de transfert

```
PORT
 200
 500, 501, 421, 530
PASV
 227
 500, 501, 502, 421, 530
MODE
 200
 500, 501, 504, 421, 530
TYPE
 200
 500, 501, 504, 421, 530
STRU
 200
 500, 501, 504, 421, 530
```

Commandes de service fichiers

```
ALLO
 200
 202
 500, 501, 504, 421, 530
REST
```

```
 500, 501, 502, 421, 530
 350
STOR
 125, 150
 (110)
 226, 250
 425, 426, 451, 551, 552
 532, 450, 452, 553
 500, 501, 421, 530
STOU
 125, 150
 (110)
 226, 250
 425, 426, 451, 551, 552
 532, 450, 452, 553
 500, 501, 421, 530
RETR
 125, 150
 (110)
 226, 250
 425, 426, 451
 450, 550
 500, 501, 421, 530
LIST
 125, 150
 226, 250
 425, 426, 451
 450
 500, 501, 502, 421, 530
NLST
 125, 150
 226, 250
 425, 426, 451
 450
 500, 501, 502, 421, 530
APPE
  125, 150
  (110)
  226, 250
  425, 426, 451, 551, 552
  532, 450, 550, 452, 553
  500, 501, 502, 421, 530
 RNFR
  450, 550
  500, 501, 502, 421, 530
  350
 RNTO
  250
  532, 553
  500, 501, 502, 503, 421, 530
```

```
DELE
 250
 450, 550
 500, 501, 502, 421, 530
RMD
 250
 500, 501, 502, 421, 530, 550
MKD
 257
 500, 501, 502, 421, 530, 550
PWD
 257
 500, 501, 502, 421, 550
ABOR
 225, 226
 500, 501, 502, 421
```

Commandes d'information

```
SYST
 215
 500, 501, 502, 421
STAT
 211, 212, 213
 450
 500, 501, 502, 421, 530
HELP
 211, 214
 500, 501, 502, 421
```

Commandes diverses

```
SITE
 200
 202
 500, 501, 530
NOOP
 200
 500, 421
```

## 6. DIAGRAMMES D'ETAT

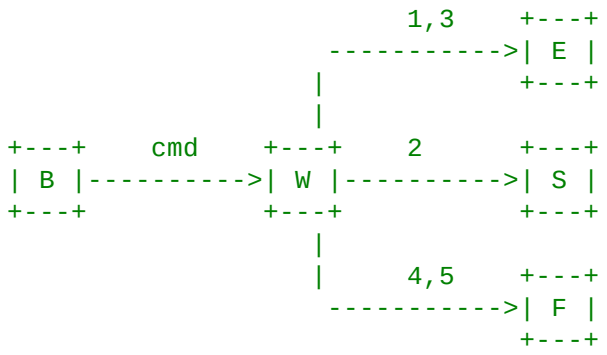
Cette section présente les diagrammes d'états pour une implémentation basique de FTP. Seul le premier digit des codes de réponse est considéré. Un diagramme d'état est associé à un groupe de commandes FTP ou à une séquence de commandes.

Le regroupement des commandes a été opéré lors de la construction du modèle de chaque commande par association des commandes qui répondaient au même modèle structurel.

Pour chaque commande, ou séquence de commandes, trois issues sont possibles: succès S (Success), Echec F (Failure), et erreur E (Error).

Dans les diagrammes ci-dessous, nous avons utilisé le symbole B pour "begin" (début), et le symbole W pour "wait for reply" (attente de réponse).

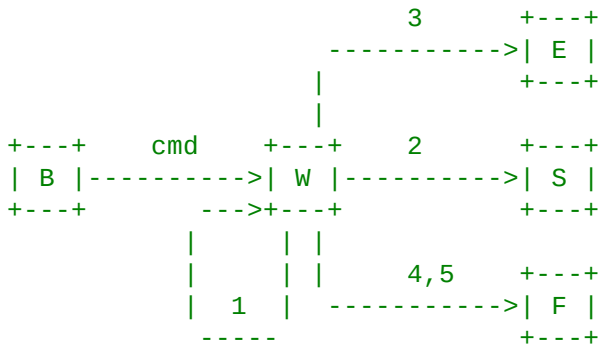
Le diagramme suivant présente le fonctionnement de la plus grande partie des commandes FTP :




Ce diagramme propose le modèle pour les commandes :

ABOR, ALLO, DELE, CWD, CDUP, SMNT, HELP, MODE, NOOP, PASV, QUIT, SITE, PORT, SYST, STAT, RMD, MKD, PWD, STRU, et TYPE.

Un autre groupe important de commande répond au schéma suivant légèrement différent :

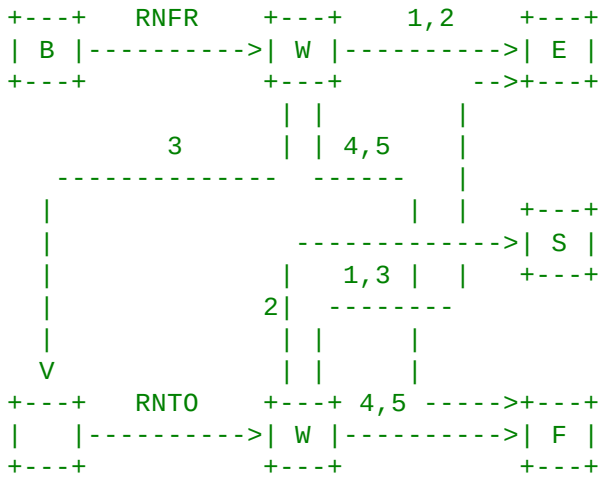



Ce diagramme propose le modèle pour les commandes :

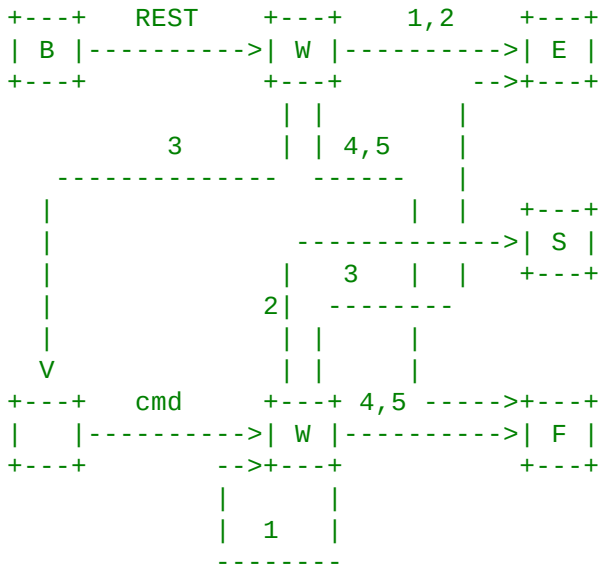
APPE, LIST, NLST, REIN, RETR, STOR, et STOU.

Notez que ce second modèle pourrait servir à représenter les commandes du premier groupe, La seule différence est que dans le premier groupe, les réponses de la classe 100 ne sont pas attendues et de ce fait sont traitées comme une erreur, tandis que le second groupe peut attendre (et parfois nécessiter) une réponse de classe 100. Une réponse au plus de classe 100 est autorisée par commande.

Les diagrammes restants donnent un modèle pour des séquences de commandes, une des plus simples est la séquence conduisant au renommage d'un fichier :



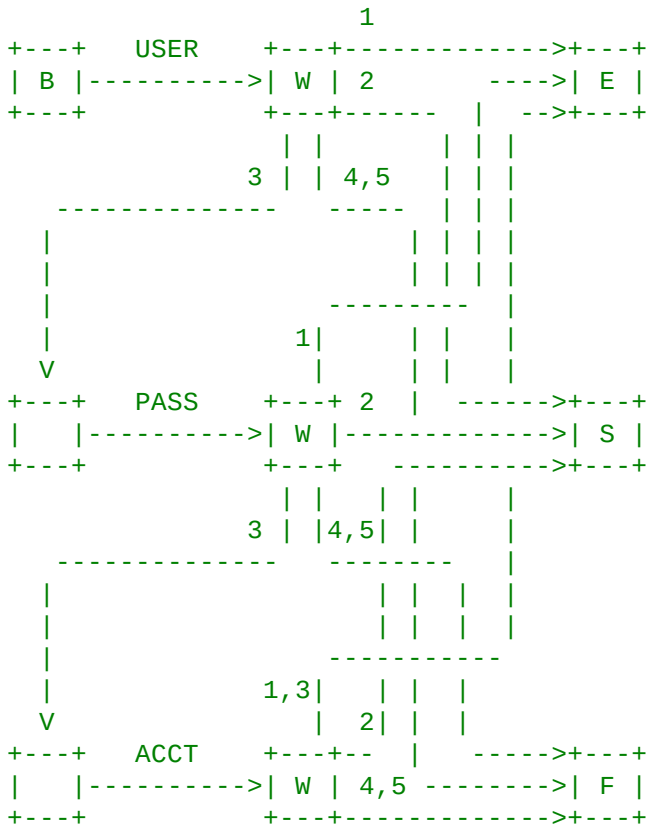

Le diagramme suivant propose un modèle simple pour une commande de reprise :



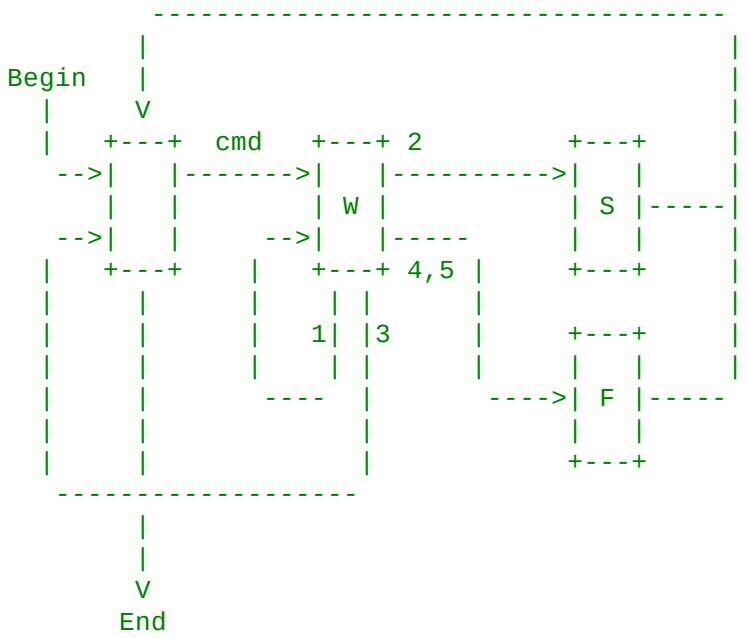

Dans lequel "cmd" est APPE, STOR, ou RETR.

Nous avons noté que ces deux diagrammes apparaissent comme très similaires. La commande de reprise diffère de celle de renommage dans le seul traitement des réponses de classe 100 du second étage, tandis que le second groupe attend (et parfois impose) une commande de classe 100. Souvenez vous qu'une seule commande de classe 100 est permise pour une commande donnée.

Le diagramme le plus complexe est pour l'ouverture d'une session (Login):




Enfin, nous présenterons un diagramme généralisé qui peut être utilisé comme modèle universel d'un échange de commandes et réponses :



## 7. SCENARIOS FTP TYPIQUES

Un utilisateur au port U voulant transférer ou recevoir des fichiers d'un serveur S:

En général, l'utilisateur communique avec le serveur via la médiation d'un processus USER-FTP. Ce qui suit peut être pris comme scénario typique. Les "prompts" USER-FTP sont montrés entre parenthèses, '---->' désigne une commande de l'utilisateur U vers l'hôte S, et '<----' désigne une réponse de l'hôte S à l'utilisateur U.

```
   COMMANDES LOCALES (Utilisateur)    ACTION IMPLIQUEE

ftp (host) multics<CR>          Connexion à l'hôte S, port L,
                                Etblissement du canal de contrôle.
                                <---- 220 Service ready <CRLF>.
username Doe <CR>               USER Doe<CRLF>---->
                                <---- 331 User name ok,
                                          need password<CRLF>.
password mumble <CR>            PASS mumble<CRLF>---->
                                <---- 230 User logged in<CRLF>.
retrieve (local type) ASCII<CR>
(local pathname) test 1 <CR>    Le USER-FTP ouvre un fichier local en
                                ASCII.
(for. pathname) test.pl1<CR>    RETR test.pl1<CRLF> ---->
                                <---- 150 File status okay;
                                      about to open data
                                      connection<CRLF>.
                                Le serveur établit le canal de données
                                vers le port U.

                                <---- 226 Closing data connection,
                                    file transfer successful<CRLF>.
type Image<CR>                  TYPE I<CRLF> ---->
                                <---- 200 Command OK<CRLF>
store (local type) image<CR>
(local pathname) file dump<CR>  Le USER-FTP ouvre le fichier local
                                sous Image.
(for.pathname) >udd>cn>fd<CR>   STOR >udd>cn>fd<CRLF> ---->
                                <---- 550 Access denied<CRLF>
terminate                       QUIT <CRLF> ---->
                                Le serveur ferme toutes les connexions
```

## 8. ETABLISSEMENT DE LA CONNEXION

Le canal de contrôle FTP est établi via TCP entre le port U du processus utilisateur et le port L de l'hôte serveur. Au protocole FTP est en effet assigné le port 21 (25 octal), c'est à dire L=21.

## APPENDICE I - STRUCTURE DE PAGE

Le besoin de FTP de supporter la structure en pages de certains fichiers est motivée par le besoin de transferts de fichiers fiables entre des systèmes TOPS-20, et en particulier les fichiers utilisés par NLS.

Le système de fichiers d'un TOPS-20 est basé sur le concept de pages. Le système d'exploitation de cette plate-forme est bien plus performant lorsqu'il manipule les fichiers sous cette forme. Le système d'exploitation fournit en général une interface vers le système de fichiers afin que des applicatifs puissent voir les fichiers comme des flux continus d'octets conformément à l'usage le plus répandu. Cependant, certaines applications utilisent directement la structure sous-jacente en pages.

Un fichier disque d'un TOPS-20 est composé de quatre éléments: un chemin d'accès, un table de pages, un ensemble de pages (peut être vide), et un ensemble d'attributs.

Le chemin d'accès est celui spécifié dans les commandes RETR ou STOR. Il inclut le nom du répertoire, le nom du fichier, l'extension de fichier, et un numéro de génération.

La table de pages peut contenir jusqu'à 2**18 entrées. Chaque entrée peut être marquée EMPTY (vide), ou peut "pointer" sur une page. Si elle est non vide, elle contiendra de plus quelques bits de marquage propres à la page; les protections d'accès sur les fichiers et sur chacune des pages peuvent être gérées indépendamment.

Une page est un ensemble de 512 mots de 36 bits chacun contigus.

Les attributs du fichier, dans le descripteur de fichier (File Descriptor Block, ou FDB), contiennent des informations telles que la date de création, la date de dernière écriture, la date de dernière lecture, la taille d'octets logique de l'auteur, le pointeur de fin de fichier, un décompte des accès en lecture et en écriture, numérotation de sauvegarde, etc.

Notez qu'il n'est pas obligatoire que les entrées soient contiguës dans la table. On pourra trouver des pointeurs de page vides entre deux pointeurs de pages réelles. De plus, la fin de fichier est simplement marquée comme un nombre. Il n'est pas obligatoire que cette fin de fichier pointe sur la dernière donnée du fichier. Des appels d'I/O séquentiels basiques sur un TOPS-20 laissera le pointeur de fin de fichier effectivement positionné après la dernière donnée inscrite, mais d'autres opérations peuvent très bien ne pas laisser le pointeur à cet endroit, si un programme particulier le souhaite.

En fait, les deux cas particuliers d'un fichier discontinu (pages vides insérées entre des pages non vides) et d'un pointeur de fin-de-fichier n'indiquant pas la dernière page du fichier, peuvent survenir dans un système géré sous NLS.

Les fichiers paginés d'un TOPS-20 peuvent être émis sous FTP en marquant les paramètres de transfert ainsi : TYPE L 36, STRU P, et MODE S (en fait, tout mode peut être utilisé).

Chaque page d'information dispose d'une en-tête. Chaque champ d'en-tête, constitué d'un mot logique, est un mot du système TOPS-20, dans la mesure où le TYPE est L 36.

Ces champs d'en-tête sont :

**Mot 0: Header Length** Longueur d'en-tête

La longueur d'en-tête vaut 5.

**Mot 1: Page Index.** Index de page

Si les données représentent une page d'un fichier sur disque, il s'agit du numéro de la page reportée dans la table de pages. Les pages vides (trous) d'un fichier ne sont tout simplement pas transférées. Notez qu'un trou n'est pas la même chose qu'une page pleine de zéros.

**Mot 2: Data Length.** Longueur des données

La longueur des données situées dans la page et après l'en-tête, exprimée en mots logiques. De ce fait, la longueur totale transmise est la somme de la longueur d'en-tête et de la longueur des données.

**Mot 3: Page Type.** Type de page

Un code pour indiquer de quel type est le segment de données. Une page de données a pour type 3, la page FDB a pour type 2.

**Mot 4: Page Access Control.** Contrôle d'accès de page

Les bits de contrôle d'accès associés à cette page de la table de pages. (Le mot complet est mis dans l'AC2 d'un SPACS par le programme lisant depuis le réseau vers le disque).

Juste après l'en-tête, on trouvera Data Length mots de données. Cette longueur est couramment de 512 pour une page de données, et de 31 pour la FDB. Les zéros de bourrage d'une page peuvent être omis, donnant ainsi une longueur de données inférieure à 512 si tel est le cas.

## APPENDICE II - COMMANDES DE REPERTOIRE

UNIX disposant d'une structure de répertoires en arbre dont les éléments (répertoires) sont simples à manipuler car compris comme des fichiers à part entière, il apparaît utile d'étendre les fonctionnalités des serveurs FTP sur ce type de plate-forme pour y adjoindre des commandes de manipulation de répertoires. Dans la mesure ou d'autres types d'hôtes de l'ARPA-Internet disposent aussi de structures de répertoires en arbre (y compris TOPS-20 et Multics), ces commandes ont été ajoutées sous la forme la plus générique possible.

Quatre commandes de manipulation de répertoire ont été adjointes à la définition de FTP :

**MKD chemin d'accès**

Crée un nouveau répertoire de nom "chemin d'accès".

**RMD chemin d'accès**

Efface le répertoire de nom "chemin d'accès".

**PWD**

Imprime le nom du répertoire de travail courant.

**CDUP**

Remonte au père du répertoire courant de travail. Le père devient le nouveau répertoire de travail.

L'argument "chemin d'accès" devra être créé (ou détruit) au titre de sous-répertoire du répertoire courant, sauf si la chaîne "chemin d'accès" contient suffisamment d'information pour qu'il en soit fait autrement, ex., "chemin d'accès" est exprimé sous forme absolue (sous UNIX et Multics), ou ce chemin ressemble à quelque chose comme "<abso.lute.path>" sous TOPS-20.

### *CODES DE REPONSE*

La commande CDUP est un cas particulier de la commande CWD, et est disponible pour simplifier l'écriture de programmes transférant des arborescences complètes entre des systèmes qui notent par une syntaxe différente l'accès au répertoire père. Les codes de réponse à une commande CDUP sont identiques à ceux obtenus pour une commande CWD.

Les codes de réponse à la commande RMD sont identiques à ceux obtenus pour son homologue "fichiers", DELE.

Les codes de réponse pour la commande MKD, par contre, sont un peu plus compliqués. Un répertoire nouvellement créé sera très fréquemment l'argument d'une commande CWD ultérieure. Malheureusement, l'argument passé à la commande MKD n'est pas toujours exploitable directement par la commande CWD. C'est le cas par exemple, lorsqu'un sous-répertoire est créé sous TOPS-20 en donnant juste le nom du sous-répertoire. C'est à dire, la séquence suivante pour un serveur FTP tournant sous TOPS-20

```
MKD MYDIR
CWD MYDIR
```

va échouer. Le nouveau sous-répertoire ne peut en effet être accédé qu'en mode "absolu"; ex., si la commande MKD avait été envoyée pour un répertoire courant <DFRANKLIN>, l'accès au nouveau sous-répertoire ne peut se faire que par la spécification de <DFRANKLIN.MYDIR>.

Même sous UNIX et Multics, l'argument passé à MKD peut aussi ne pas convenir. S'il s'agit d'un chemin "relatif" (c-à-d., un chemin interprété à partir du répertoire courant), l'utilisateur devra nécessairement être positionné dans ce répertoire initial pour atteindre le sous-répertoire considéré par cette expression. Suivant l'application,

cela peut se révéler inconfortable. Dans tous les cas, ce principe manque de robustesse.

Pour résoudre ces problèmes, et sur conclusion positive d'une commande MKD, le serveur devrait retourner de préférence la chaîne suivante :

```
257<espace>"<nom-répertoire>"<espace><commentaire>
```

Le serveur, par là, indique quelle est la chaîne à utiliser pour accéder au répertoire nouvellement créé. Le nom de répertoire peut contenir n'importe quels caractères; un guillemet dans la chaîne devra être doublé.

Par exemple, un utilisateur se branche sur le répertoire /usr/dm, et crée un sous répertoire, appelé "chemin":

```
CWD /usr/dm
200 directory changed to /usr/dm
MKD chemin
257 "/usr/dm/chemin" directory created
```

Voici un exemple avec un guillemet "dans le texte" :

```
MKD foo"bar
257 "/usr/dm/foo""bar" directory created
CWD /usr/dm/foo"bar
200 directory changed to /usr/dm/foo"bar
```

L'existence préalable d'un sous répertoire de même nom est considéré comme une erreur, et le serveur doit renvoyer une erreur "access denied".

```
CWD /usr/dm
200 directory changed to /usr/dm
MKD chemin
521-"/usr/dm/chemin" directory already exists;
521 taking no action.
```

Les réponses d'erreur pour MKD sont analogues à son homologue "fichiers", STOR. De même, une réponse "access denied" est donnée si le nom du répertoire entre en conflit avec un autre fichier existant pour ce chemin, quel que soit son type (ceci est effectivement un problème d'UNIX, mais ne devrait pas en être un sous TOPS-20).

Comme la commande PWD renvoie le même type d'information que la commande MKD pour une conclusion positive, le code de réponse positive pour la commande PWD utilisera également le code 257.

### *SUBTILITES*

Comme ces commandes seront les plus utiles pour transférer des arborescences d'une machine à l'autre, il faudra se rappeler que l'argument donné à MKD doit être interprété comme un chemin relatif par rapport au répertoire courant, sauf si ce chemin

contient suffisamment d'information pour qu'il en soit autrement. Voici un exemple hypothétique sur un système TOPS-20 :

```
CWD <some.where>
200 Working directory changed
MKD overrainbow
257 "<some.where.overrainbow>" directory created
CWD overrainbow
431 No such directory
CWD <some.where.overrainbow>
200 Working directory changed

CWD <some.where>
200 Working directory changed to <some.where>
MKD <unambiguous>
257 "<unambiguous>" directory created
CWD <unambiguous>
```

Notez que le premier exemple se termine par la création d'un sous répertoire du répertoire courant d'accès. Par contre, l'argument donné dans la deuxième création fournit assez d'information au système TOPS-20 pour que celui-ci puisse considérer de façon univoque que le répertoire à créer doit l'être à partir de la racine. Notez d'ailleurs dans le premier exemple, que l'utilisateur à "violé" le protocole en essayant d'accéder directement au répertoire nouvellement créé sans utiliser la chaîne retournée par le système TOPS-20. Le problème aurait pu être masqué par l'existence d'un répertoire <overrainbow> au niveau haut de hiérarchie; ceci est une ambiguïté propre à certaines implémentations du système TOPS-20. Des considérations similaires s'appliquent à la commande RMD. Le cas est le suivant : sauf là où cette manipulation violerait des conventions internes du système pour la description de chemins d'accès relatifs et absolus, l'hôte doit traiter les arguments des commandes MKD et RMD comme des sous répertoires relatifs. La réponse de code 257 à une commande MKD doit toujours renvoyer le chemin d'accès absolu nouvellement créé.

## APPENDICE III - RFC à propos de FTP


Bhushan, Abhay, "A File Transfer Protocol", RFC 114 (NIC 5823), MIT-Project MAC, 16 April 1971.

Harslem, Eric, and John Heafner, "Comments on RFC 114 (A File Transfer Protocol)", RFC 141 (NIC 6726), RAND, 29 April 1971.

Bhushan, Abhay, et al, "The File Transfer Protocol", RFC 172 (NIC 6794), MIT-Project MAC, 23 June 1971.

Braden, Bob, "Comments on DTP and FTP Proposals", RFC 238 (NIC 7663), UCLA/CCN, 29 September 1971.

Bhushan, Abhay, et al, "The File Transfer Protocol", RFC 265

(NIC 7813), MIT-Project MAC, 17 November 1971.

McKenzie, Alex, "A Suggested Addition to File Transfer Protocol", RFC 281 (NIC 8163), BBN, 8 December 1971.

Bhushan, Abhay, "The Use of "Set Data Type" Transaction in File Transfer Protocol", RFC 294 (NIC 8304), MIT-Project MAC, 25 January 1972.

Bhushan, Abhay, "The File Transfer Protocol", RFC 354 (NIC 10596), MIT-Project MAC, 8 July 1972.

Bhushan, Abhay, "Comments on the File Transfer Protocol (RFC 354)", RFC 385 (NIC 11357), MIT-Project MAC, 18 August 1972.

Hicks, Greg, "User FTP Documentation", RFC 412 (NIC 12404), Utah, 27 November 1972.

Bhushan, Abhay, "File Transfer Protocol (FTP) Status and Further Comments", RFC 414 (NIC 12406), MIT-Project MAC, 20 November 1972.

Braden, Bob, "Comments on File Transfer Protocol", RFC 430 (NIC 13299), UCLA/CCN, 7 February 1973.

Thomas, Bob, and Bob Clements, "FTP Server-Server Interaction", RFC 438 (NIC 13770), BBN, 15 January 1973.

Braden, Bob, "Print Files in FTP", RFC 448 (NIC 13299), UCLA/CCN, 27 February 1973.

McKenzie, Alex, "File Transfer Protocol", RFC 454 (NIC 14333), BBN, 16 February 1973.

Bressler, Bob, and Bob Thomas, "Mail Retrieval via FTP", RFC 458 (NIC 14378), BBN-NET and BBN-TENEX, 20 February 1973.

Neigus, Nancy, "File Transfer Protocol", RFC 542 (NIC 17759), BBN, 12 July 1973.

Krilanovich, Mark, and George Gregg, "Comments on the File Transfer Protocol", RFC 607 (NIC 21255), UCSB, 7 January 1974.

Pogran, Ken, and Nancy Neigus, "Response to RFC 607 - Comments on the File Transfer Protocol", RFC 614 (NIC 21530), BBN, 28 January 1974.

Krilanovich, Mark, George Gregg, Wayne Hathaway, and Jim White, "Comments on the File Transfer Protocol", RFC 624 (NIC 22054), UCSB, Ames Research Center, SRI-ARC, 28 February 1974.

Bhushan, Abhay, "FTP Comments and Response to RFC 430", RFC 463 (NIC 14573), MIT-DMCG, 21 February 1973.

Braden, Bob, "FTP Data Compression", RFC 468 (NIC 14742), UCLA/CCN, 8 March 1973.

Bhushan, Abhay, "FTP and Network Mail System", RFC 475 (NIC 14919), MIT-DMCG, 6 March 1973.

Bressler, Bob, and Bob Thomas "FTP Server-Server Interaction - II", RFC 478 (NIC 14947), BBN-NET and BBN-TENEX, 26 March 1973.

White, Jim, "Use of FTP by the NIC Journal", RFC 479 (NIC 14948), SRI-ARC, 8 March 1973.

White, Jim, "Host-Dependent FTP Parameters", RFC 480 (NIC 14949), SRI-ARC, 8 March 1973.

Padlipsky, Mike, "An FTP Command-Naming Problem", RFC 506 (NIC 16157), MIT-Multics, 26 June 1973.

Day, John, "Memo to FTP Group (Proposal for File Access Protocol)", RFC 520 (NIC 16819), Illinois, 25 June 1973.

Merryman, Robert, "The UCSD-CC Server-FTP Facility", RFC 532 (NIC 17451), UCSD-CC, 22 June 1973.

Braden, Bob, "TENEX FTP Problem", RFC 571 (NIC 18974), UCLA/CCN, 15 November 1973.

McKenzie, Alex, and Jon Postel, "Telnet and FTP Implementation - Schedule Change", RFC 593 (NIC 20615), BBN and MITRE, 29 November 1973.

Sussman, Julie, "FTP Error Code Usage for More Reliable Mail Service", RFC 630 (NIC 30237), BBN, 10 April 1974.

Postel, Jon, "Revised FTP Reply Codes", RFC 640 (NIC 30843), UCLA/NMC, 5 June 1974.

Harvey, Brian, "Leaving Well Enough Alone", RFC 686 (NIC 32481), SU-AI, 10 May 1975.

Harvey, Brian, "One More Try on the FTP", RFC 691 (NIC 32700), SU-AI, 28 May 1975.

Lieb, J., "CWD Command of FTP", RFC 697 (NIC 32963), 14 July 1975.

Harrenstien, Ken, "FTP Extension: XSEN", RFC 737 (NIC 42217), SRI-KL, 31 October 1977.

Harrenstien, Ken, "FTP Extension: XRSQ/XRCP", RFC 743 (NIC 42758), SRI-KL, 30 December 1977.

Lebling, P. David, "Survey of FTP Mail and MLFL", RFC 751, MIT, 10 December 1978.

Postel, Jon, "File Transfer Protocol Specification", RFC 765, ISI, June 1980.

Mankins, David, Dan Franklin, and Buzz Owen, "Directory Oriented FTP Commands", RFC 776, BBN, December 1980.

Padlipsky, Michael, "FTP Unique-Named Store Command", RFC 949, MITRE, July 1985.

## REFERENCES

[1] Feinler, Elizabeth, "Internet Protocol Transition Workbook", Network Information Center, SRI International, March 1982.

[2] Postel, Jon, "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, DARPA, September 1981.

[3] Postel, Jon, and Joyce Reynolds, "Telnet Protocol Specification", RFC 854, ISI, May 1983.

[4] Reynolds, Joyce, and Jon Postel, "Assigned Numbers", RFC 943, ISI, April 1985.