MDFS software version 0.AA Release Notes

New features

This release supports :

- Partitioning of large hard discs into 35 Mbyte logical discs.
- Up to 4 hard discs or partitions (formerly 2).
- Retrieval of individual files from tape.
- 2048 accounts (numbered 000 to 7FF). See separate notes.

Important change

It is important to *park* the heads of hard discs if the equipment is to be moved. Previous versions of the fileserver parked the heads whenever the release discs button was pressed. Some modern disc drives will shut down completely when instructed to park the head, which is undesirable when simply changing floppy discs.

The new software will only park the heads if the release discs button is *pressed and held for five seconds* when the fileserver is running. This will cause the discs free lamp to illuminate steadily, indicating that the fileserver is completely shut down. When changing discs, the button should just be pressed momentarily, which will give the usual flashing discs free/online lamps.

Alternatively, the heads may be parked with the **Z** command in utility mode.

You are warned that transporting hard disc drives without having parked the heads can cause **permanent damage** to the surface of the disc.

Partitioning of large discs

Hard discs with a capacity greater than 35 Mbytes are now divided into *partitions*, each of which functions as if it were a separate disc drive. Each partition has a capacity of 35 Mbytes, except the last partition on each disc which will use up any remaining space on the disc. For example, a 95 Mbyte disc will have two partitions of 35 Mbytes plus one partition of 25 Mbytes. The fileserver can only support a total of 4 partitions when online. If discs with a total of more than 4 partitions are connected, the fileserver will use the first partition on each disc, plus the largest partitions available from the other discs up to the maximum of 4. Hence with a 95 Mbyte disc (35+35+25 partitions) plus a 37 Mbyte disc (35+2) the 2 Mbyte partition will be ignored. Four floppy discs can always be supported in addition to any hard discs.

When the fileserver is online, the separate partitions function as independent discs and are referenced by name. Use *FREE to list the available discs/partitions.

In utility mode, the discs are referred to by letter (corresponding to the hardware controller number & drive number). If a large disc is selected, the system will prompt for a partition number (key 1 for the first partition, 2 for the second, up to the number of partitions on that particular disc). In the case of the Verify command, individual partitions can be verified or the whole disc can be verified at once by specifying a partition number of zero. Use the **L** (list discs) command to see the partitions available on all discs.

When using a tape drive, each tape can hold the contents of one disc partition. It is therefore useful to arrange your files on the disc such that frequently updated files are in one partition which can be backed up daily, leaving constant material (such as software packages, archive material, read-only databases etc.) on another partition to be backed up less frequently.

Initialising existing hard discs

Customers with discs larger than 35 Mbyte which have been used with earlier versions of the fileserver will only have been able to use the first partition on the disc. When the new software is installed, the other partitions will still not be available as the root directories will not have been created when the disc was formatted - messages such as *Block 0 corrupt on drive E2* will be produced on the printer.

To create these directories, either the disc must be formatted again, or the roots of each partition must be initialised. The format command now prompts for separate disc names for each partition. Beware that formatting a disc *erases all data from all partitions on that disc.* Only format a disc if you are happy to lose the data stored on that disc.

Chair Green

To initialise a partition without corrupting other data on that disc, the **I** (initialise disc) command may be used (in utility mode). Note that this command does not appear on the menu which is displayed, but works just like the commands that do appear on the menu. Before using the **I** command, type **L** to obtain a list of the discs and partitions available. Partitions which need initialising will say *Not a fileserver disc* in place of the disc name - on discs formatted with the old software, partition 1 will contain the existing data, while partitions 2, 3 etc will need initialising. Now type **I** and enter the drive letter followed by the partition number. The current name of the disc will be displayed (this should say *Not a fileserver disc* if the partition needs initialising), and the program will ask for a name for that partition. Enter a name which is different from any other discs or partitions in your system. Finally you will be asked to type Y or N to confirm that the details are correct - type **Y** if you are sure that you have specified the correct partition. Take care **not** to initialise partition 1 as that would erase the data previously stored on the disc. Repeat the **I** command if there are more partitions to initialise.

Retrieval of individual files from tape

It is now possible to read backup tapes without restoring the whole tape onto a disc. This is particularly useful for recovering files which have accidentally been deleted.

The facility works while the fileserver is online, by making the tape appear as if it were a very slow disc drive. It is not possible to write to the tape in this mode.

If a tape is inserted in the tape drive while the fileserver is online, a special directory **%TAPE** becomes available. This is equivalent to the root ($) directory on the disc that was backed up onto the tape. Hence the following might be used to recover a BASIC program :

```
>*I AM FRED
>*DIR %TAPE
>*DIR FORM3
>*DIR FRED
>*CAT
FRED        (073)       Owner
ARG1                    Option 00 (Off)
Dir. FRED               Lib. LIBRARY

Bescfix     WR/r        BFASTCOMP   WR/wr       Bincode     WR/r        BMINITERM   WR/r
Bsafeterm   WR/r        Bxmit       WR/r        CARDS       D/          cst         D/

>LOAD"Bxmit"
>*DIR
>SAVE"OldXmit"
>*UNLOADTAPE
>
```

No privilege is required to access the tape; the files on tape still have account numbers and access letters attached, so access is controlled in just the same way as files on the main disc. All the usual commands (eg. **CAT, *EX, *INFO*) can be used to inspect the contents of the tape. The utilities *Copier* and *Multicopy* can be used to copy groups of files.

One problem with this system is the slow response of the tape drive, which can cause *No Reply* errors. If such an error occurs, wait for the tape to finish winding and repeat the sequence of commands. The necessary data should now be held in memory in the fileserver and so the commands will succeed immediately. The following guidelines will minimise the possibility of these errors :

- Choose a time when there are as few people as possible using the fileserver, as other users will use up valuable memory space.
- Note that the example above selected the directory in a series of steps, rather than ***DIR %TAPE.FORM3.FRED** which would have required the fileserver to do all the work in the time allowed for one operation. Always divide up long pathnames in this way.
- Use *Copier* or *Multicopy* on a BBC Master (or ET or Compact). This combination allows longer for each operation.

Before removing the tape, it should be *unloaded* to protect the surface of the tape from contamination. To do this, either press the *Release Discs* button, or use the *UNLOADTAPE command. When the tape has finished winding, it may be removed by pressing the large button under the tape slot.

Fileserver software with 2048 accounts

Introduction

The fileserver software for HDFS and MDFS has been enhanced to allow the use of 2048 account numbers, with effect from version 0.AA . The new system is upwards-compatible from the old; no special action is required when installing the new software onto existing discs, but once the enhanced facilities have been used on a disc, that disc should not be used with earlier versions of the fileserver.

A new version of EDITPASS is provided to allow access to these accounts. In addition, a suite of programs for managing very large password files is now available - see the separate documentation.

Availability of account numbers

The new accounts are identical to the old in most respects; a separate balance can be kept for each account on each disc and files can be freely assigned to any account. The main difference is that there are now some restrictions on the way in which account ownership can be given to users.

Under the old system, users could be given ownership of any combination of accounts. However, each user would typically have one personal account in which to keep all their own files, while some users would share ownership of a few more accounts giving access to printers, or for shared project work. In addition, the system manager and other users in positions of authority would be granted ownership of other users' personal accounts for supervisory purposes.

The new system formalises this pattern of use. The system now records a personal account for each user, which may be freely chosen from the whole range of account numbers (000 to 7FF), but any shared accounts must have numbers between 000 and 0FF. For the system manager and other 'super users', it is possible to have access to more than one account above 0FF, but only in blocks of 64 accounts. For example, a class of pupils might have personal accounts 100, 101, 102 ... 11A. To give the class teacher ownership of the pupils' files, it would be necessary to give him ownership of accounts 100 to 13F. See below for the available blocks of account numbers.

The usual allocation of account numbers on a large system is to allocate all users' personal accounts between 100 and 7FF, starting each group of users on a multiple of 64 (100, 140, 180 etc.) so that super users can be given blocks of accounts which match up with the users that they are to supervise. Accounts 000 to 0FF are then available for shared use. Note that 000 is usually the account number of the root ($) directory, and so should only be owned by the system manager. It is customary to reserve a few more accounts (say 000 to 01F), leaving the remainder up to 0FF for shared use.

There is no restriction on the use of personal account numbers, so on a small system personal accounts could start from, say, 020 and accounts above 0FF need not be used at all.

If the personal account is set to zero, ownership of account 000 is *not* granted, and the user effectively has no personal account. This is useful for public users, such as BOOT or ANONPRINT.

Note that when a user is printing, the job in the print queue is given an auxiliary account number equal to the user's personal account number, or if the user has no personal account equal to the highest numbered account that the user owns, or zero if the user owns no accounts at all. This is so that each user has ownership of his own print jobs, which is necessary to delete, reroute, or flush them.

Storage of account balances

To store the current balance for all accounts requires 4K of disc space on each disc. For small systems where not all accounts are in use, particularly with floppy discs, this is an unnecessary overhead. The system therefore does not allocate this space initially, and so several account numbers share the same balance; account 000 has the same balance as 100, 200, 300 etc. If a large number of accounts are in use, the fileserver can be instructed to take up the necessary disc space and the accounts will have independent balances.

`*MAXACC 7FF`	Uses 4K of disc space, and all accounts have independent balances.
`*MAXACC 1FF`	Uses 1K of disc space, accounts 000-1FF all have independent balances, accounts 2xx, 3xx, 4xx, 5xx, 6xx, 7xx all share balances.

Note that *MAXACC only applies to the currently selected disc; to reserve balances on all discs, it is necessary to select the disc with *DIR (or *SDISC) and then use *MAXACC, repeating for all the discs on the system.

Users of hard disc systems would usually use *MAXACC 7FF on all discs and use the accounts independently. With floppy discs, it is rarely necessary to have more than 256 accounts on each disc, but users on separate discs cannot share an account number as their files need to be protected from each other. If each floppy disc is allocated a range of accounts, say 110-1FF, 210-2FF etc., and other accounts are not used on that disc, each account will still have an independent balance (because balances are maintained separately on each disc) at no extra cost in disc space.

The effect of *MAXACC is permanent; once allocated, the disc space cannot be recovered without re-formatting the disc.

Upgrading from earlier versions of fileserver software

The new software should be installed in the usual way (*TPOKER* on HDFS, or by copying $.FS on MDFS). In the case of MDFS, care should be taken to copy the new version of $.FS to all discs which are used to start the fileserver - especially where floppy discs are used. The only change which will be apparent at this stage is that account numbers in *INFO or *EX now have three digits; no users (even *SYST*) will own the new accounts, so the system will continue to operate as before.

No action is required if the new accounts are not used; the new software operates identically to the old.

To start using the new accounts, it is necessary to use the new version of EDITPASS, initially to give *SYST* ownership of accounts 100-7FF. When existing users are inspected with the new EDITPASS they will have no personal account, but the account which they have been using as a personal account will appear as one of their shared accounts. There is no need to change this, as all users will have access to their files, but in due course it may be desirable to rationalise existing allocations of account numbers to fit in with the new scheme.

In most cases, it will be necessary to use *MAXACC before allocating any new accounts to users. Note that when the account balances are separated by *MAXACC they are not bankrupted, but each account will have the same balance as those which it was formerly sharing balance storage. Care must be taken to give new users an appropriate starting balance, or all the balances can be zeroed to avoid future mistakes :

```
10 FOR A%=&100 TO &7FF
20 OSCLI"DEBIT "+STR$~A%+" 65535"
30 NEXT
```

(Note that this requires BASIC 2 or better)

Available blocks of high numbered accounts.

Users who need ownership of more than one account with numbers greater than 0FF may be given ownership of one or more of the following blocks. If an attempt is made to give ownership of a range which does not fit exactly onto these blocks, EDITPASS will allocate enough blocks to cover the whole of the specified range, which will give ownership of more accounts than were actually specified.

100-13F	140-17F	180-1BF	1C0-1FF	200-23F	240-27F	280-2BF	2C0-2FF
300-33F	340-37F	380-3BF	3C0-3FF	400-43F	440-47F	480-4BF	4C0-4FF
500-53F	540-57F	580-5BF	5C0-5FF	600-63F	640-67F	680-6BF	6C0-6FF
700-73F	740-77F	780-7BF	7C0-7FF				

FDFS compatibility

The features of this new software will *not* be available on the FDFS. MDFS users who insert FDFS discs for data interchange should avoid using the new facilities on FDFS format discs. In particular, *MAXACC should not be used, and when editing the password file personal accounts and account numbers greater than 0FF should not be used.

Preliminary User Documentation for Password Management System

The new password file editor edits password files by converting them to a human readable ASCII text file which is then processable on any text editor. Additions and changes to the existing password file will be made by composing a text file in ASCII which is then merged with the text file that has been generated from the ASCII version of the %PASSWORDS file. This document outlines the format of these ascii password files.

There are two basic forms that these text files can take. The first is a file that contains details of modifications that are needed, eg adding new users or modifying existing entries, this is a "Mod-file". The second is a file which is generated from an existing password file, or from the merger of a mod-file and a file generated from an existing password file, this second type of file will never contain modification instructions only entries for users and is called the "Gen-file".

Format of data for individual entries

The data about each user entry is held in two different ways, as data "Local" to that particual entry and as data "Global" to a range of entries (or the whole file). Particular information such as URD or account ownership is specified by assigning two a fixed keyword of which there are two types Global and Local. The Global keywords are as follows:

ACC	For normal accounts, and for blocks of personal accounts.
BASE	For the base of the URD to which the UID is added.
FLAG	Which contains the different possible flags as two letter symbols.
LIB	The library path.
PASS	The users password.
BOOT	A default boot option.
PACC	The user's personal account {Used only in Mod-files for setting start of search for free personal account numbers}

The Local keywords are:

ACC	As above.
URD	A full URD.
FLAG	As above.
LIB	As above.
PASS	As above.
BOOT	As above.
DEFAULT	Indicates the default user. In order to unset a current default user DEFAULT must be given the value "0".
PACC	Personal account. Set to "" when no personal account is required.

Assignments are made to keywords as folows:

Keyword = "data";

eg ACC = "1,2,3"; or BASE = "$.form3";

Sometimes it is necessary to cancel the effect of a keyword or undefine it, this is done by:

Keyword = UNDEF;

eg PASS = UNDEF;

UNDEF must be in upper case.
Note that there are no quotes

In most cases the data assigned to a keyword is obvious. For accounts ">" is used to indicate a range and "-" to indicate removal of particular accounts, and "+" for addition. {Note that "+" and "-" only take effect in modify mode}. For the FLAG keyword assignment data is in the form of two

letter combinations which are as follows:

Pw	Password locked.
Sy	System.
Ns	No short saves.
En	*Enable required.
Nl	No library.
Ro	Run only user.
X1	Reserved
X2	"

If an option is preceded by a "-" it is removed and by a "+" it is added to the current list of FLAGS in use. If neither "+" or "-" is given then the value of FLAG is used directly. {At present the use of "+" and "-" only work in "MODIFY" mode}.

Global and Local keywords are distingushed from one another by the use of { and }. These curly brackets follow the UID and all keywords contained within are taken as Local to that user. Thus the form for a user entry is:

Global assignments
UID { local assignments }

Both the Global and Local assignments are optional, however the curly brackets must always be present. It is recomended that where the local assigments flow over one line that space is left under the UID so that it stands out on the page. However this is not a requirement. Thus a well formed user entry might look like this:

```
ACC="1,2,3"; BASE="$.form3"; FLAG="SaEn";

ARG          { PASS="Wombat"; LIB="$.SJLIB"; ACC="0>FF"; FLAG="Sy";
               PACC="1FF"; BOOT="3"; }
```

{In the future the quotes will be optional execept for assignment to the PASS keyword.}

There are a variety of different ways in which this entry might be used. These are termed "modes" and there are three different modes as follows:

.Add.

In this mode the user entries are taken as new users. If a user of this name already exists an error is generated. In the case of a correct addition a personal account is automatically assigned (unless the PACC keyword has been assigned to explicity), and a set of instructions placed in the file "!makedir", that can be *EXEC'ed to create the directory structure for all the new users. If MERGE is used repeatedly on the same file this continues to add the commands to create new user directories to the existing !makedir. Therefore it is important that the !makedir is deleted after use.

.Remove.

The specified users are removed from the password file. Obviously no global assignments or local assignments are needed, however it is not an error for these to exist. This makes it possible to remove blocks of users and later restore them just by changing the mode information for those entries.

{ A future addition will be the option to remove the users files and generaly tidy up the directories, this will be done as folows:

```
games       {REMOVE="1";}

}
```

.Modify.

The data in the user entries is used to supliment or modify the data already held in an existing entry. It is an error for the user not to already exist. In this mode the undefining of a keyword is necessary. Because if a Global keyword is set it must be possilbe to undefine it, so as to allow subseqent user entries to retain their old values for this particular keyword. Local keywords may also be "UNDEF"ed to allow the current global to be ignored for that user.

These three modes only apply to the mod-file and will not be encountered in the gen-file. To select a particular mode a line is entered into the mod-file before the user entries to which it refers as follows:

.mode.

Global assignments
UID { local assignments }
UID { " " }

.mode.

UID { local assignments }
UID { " " }

The possible modes are:

.Add.
.Remove.
.Modify.

In the mod-file there is no defult mode so one must be set before any UIDs are given, otherwise there will be an error. The different modes will cause different effects if an incomplete user entry is given, ie not all keywords are assigned either globaly or localy. For Modify this implies that the data from an existing entry is to be used. For Remove no notice is taken of the data. For Add sensible defaults are assumed as follows:

ACC	No accounts unless PACC is in the range 0-FF.
URD	Taken as $.<uid>
PASS	No password
LIB	Standard library.
BOOT	Taken to be off.
FLAG	Taken to be no flags set.
PACC	Assigned a sensible unique value.

Pseudo Modes

There are further modes that are used to control the flow of processing The first is ".END.", it is used in both Mod-files and Gen-files to indicate the end of the useful data in the file. Its use is optional. One use for it is to cause the remainder of a file to be ignored by one of the processing programs, though this is of little value in everyday use.

The second of these is .users xx. that informs the password file generator how many users there are to be in the file. This is inserted automatically in genfiles; the user need only be concerned with it if editing genfiles or other intermediate files directly rather than by merging in mod files.

Further notes

Although a Local keyword exists for PACC it is assumed that personal accounts will be assigned sensible values automaticaly. This is possible because during the first part of the edit process, when the ASCII file is generated from the existing password file, a map of used personal accounts are

generated. From which suitable values are chosen from unused personal account numbers.

Because of the automatic generation of personal accounts there are a number of special effects that are generated by particular settings of the PACC keyword. If a global PACC is set then searching for personal accounts for new users will start from that value. THis continues until it is unset by setting the Global PACC keyword to UNDEF, at which point the search for personal account numbers will resume at the lowest free personal account numbers. (Setting the Local PACC keyword to UNDEF will also switch the search, but only for that user) Personal accounts start at &100, however it is possible to set PACC to values below this, if this is done a warning is generated, but the assignment will take place. If a PACC is set to "" or to "0" then this is taken as meaning unset the personal account, again a warning is generated, and the personal account is unset. For newusers where the PACC has been explicitly set to "0" there is a problem in deciding which account to give to the URD, in this case the highest account number from the local ACC keyword is used, and in the case of there being no local ACC no access to the URD will be given to that user. (The reason that only the local keyword is scanned is so that account ownership of common accounts, such as PRINTQ may be given to a user with no file access).

Any line begining with a "&" is taken as a comment and totally ognored.

Program suite

The programs are run in the following order:

```
CHAIN "CONVERT"   This converts the %PASSWORDS file to the Gen-file
CHAIN "PARSE"     This parses the Mod-file
CHAIN "SORT"      This sorts the products of PARSE
CHAIN "MERGE"     This merges the Gen-file and the product of SORT
CHAIN "GENERATE"  This generates the new PASSWORDS file
*RENAME PASSWORDS %PASSWORDS
```

The use of CONVERT is self explanatory, it will need to be used each time a modification is to be made to an existing password file. The PARSE program will be the most used, as this checks the user generated Mod-file syntactically, and produces appropriate error messages. Its output is placed in the TEMP sub-directory. This is a text file that has had all the GLOBAL assignments removed, and all the PACCS for new users inserted. This file is then sorted by SORT into alphabetic order ready for merging by MERGE. MERGE processes the Mod-file and Gen-file. It is at this stage that new users are checked for name clashes with existing users, and that users specified for modification actually exist. The final product of MERGE is a text file called "passtext".

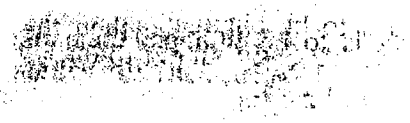
MERGE also generates a file called "!makedir". This contains execable *commands to generate the directory structures for the newusers created during the merge process. Finaly if all has gone according to plan the GENERATE program is used to create the new password file, this is called "PASSWORDS" and is generated in the current working directory. In order to replace the existing %PASSWORDS with this new file the following command should be used:

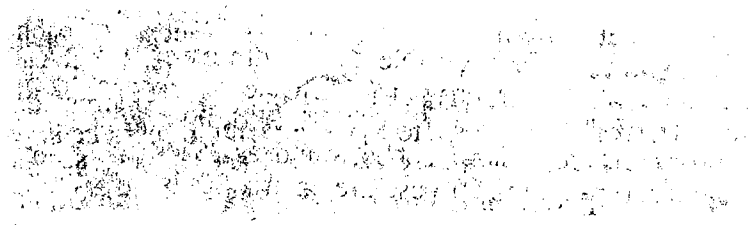
```
*RENAME PASSWORDS %PASSWORDS
```

This is the only way that a new password file should be installed.

It is important to always run SORT after PARSE, and to run PARSE each time the Mod-file is changed. There is one important restriction on the size of the Mod-file, that is that they cannot contain more than 256 users, this should not present a problem as MERGE can be used repeatedly on the same file.

The whole process described above can take some time to complete, and it is envisaged that it will only be used when making large numbers of additions or modifications to a password file. For everyday usage the normal program "EDITPASS" should be used. Where there are too many users





for this program to work, or information about personal accounts needs to be changed then the program QEDIT must be used, this has a user interface very similar to EDITPASS, but only works on a single user at a time, thus not needing to store the password file in local RAM.

In order to install the suite of programs there is an execable file called "!install" that creates a suitable directory called "pwmanage" in which the programs can reside.

Further technical information

Since all of the passwords are displayed in ASCII text in the files, it is very important that only the system manager has access to them, and they should be treated with as much secure respect as %PASSWORDS itself. Each of the programs does its best to stop unorthorised people being able to see the files, but as always security is only as good as the system manager. It is intended that in future releases of the software passwords will be encrypted.

All of the programs can run on DFS rather than network, however the variable "net%" in each program must be made FALSE for this to work correctly. This variable along with other "tweeks" can be found after the comments in the first few lines of the programs. In order to suppress all output, except errors and warnings, the variable "test%" must be set to FALSE. Should there be any need to change the filenames used by each program, these can also be found assigned to string variables within the first few lines of the code in each program. If these are changed it is important to change the names in all of the programs.

Formal definition of file spec

<file> ::= <gen-file> | <mod-file>
<gen-file> ::= [<userdata>] | [<global assignment>] [<userdata>]
<mod-file> ::= .<mode>. <gen-file>
<mode> ::= Add | Modify | Remove | End
<global assignment> ::= <global keyword> = "<keyword value>";
<userdata> ::= <Uid> | <Uid> { [<local assigment>] }
<local assignment> ::= <local keyword> = "<keyword value>";
<Uid> ::= [<alphanum>]
<global keyword> ::= ACC | LIB | PASS | BOOT | BASE | FLAG
<local keyword> ::= ACC | LIB | PASS | BOOT | URD | FLAG | PACC | DEFAULT
<keyword value> ::= <acc> | <lib> | <pass> | <boot> | <urd> | <flag> | <pacc>
<defult> | <base> | UNDEF
<acc> ::= [<hex>, | <hex> '>' <hex>, | - <acc>, | +<acc>]
<lib> ::= <path>
<pass> ::= <alphanum>
<boot> ::= 0 | 1 | 2 | 3
<urd> ::= <path>
<pacc> ::= <bighex>
<default> ::= 1 | 0
<base> ::= <path>
<flag> ::= [<flagsymbol> | +<flagsymbol> | -<flagsymbol>]
<flagsymbol> ::= Sy | Ns | Ro | Nl | En | Pa | X1 | X2
<path> ::= [<name>.] | $<discname>.<path>
<discname> ::= <alphanum>
<name> ::= <alphanum>
<hex> ::= <hexit> | <hexit><hexit>
<bighex> ::= <hex> | <hexit><hexit><hexit>
<hexit> ::= 0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F

Note there is no case sensitivity, as every alphanum is taken as upper case (?? **UNDEF** seems to be an exception). Forms shown are only preferred forms.