MenuFinder

Version 3.2

For the OpenVMS operating system

User's Guide

i3 – Italian Internet Information

NOTE: MenuFinder is packaged for distribution using Info-ZIP's compression utility. The installation program uses UnZip to read zip files from the distribution kit CD. Infor-ZIP's software, (Zip, UnZip and related utilities) are free and can be obtained as source code or executables from various bulletin board services and anonymous – fpt sites, including CompuServe's IBMPRO forum and ftp.uu.net:/pub/archiving/zip/*

First edition: September 1992

1. INTRODUCTION

1.1. What Is MenuFinder

MenuFinder is a menu generator for OpenVMS systems (VAX and AXP).

MenuFinder has been created to give a simple but effective answer to the requirement of presenting the user with a set of functions via a menu interface.
In particular, MenuFinder is an ideal instrument to

- build on-line guides for the users of the computing centre
- document and utilise management procedures of the system manager or the operator.

1.2. Main Characteristics

Types of option in the menus

DCL commands, text files, data files, programs, batch files and submenus can be associated with the options.

Association of programmes or commands to files.

For example, the program used to write a document can be associated with the latter: choosing the document from the menu, activates that program to read or write the document.

Help text on the menu options

A user may have to use many programs, although infrequently: it is not easy therefore, to remember even the simplest functions of a product. Often, shortcuts and contrivances are forgotten after a while. MenuFinder allows the possibility to associate a help text with each option of a menu, in which can be inserted instructions, annotations and useful indications both for oneself and other users.

The means of accessing the menus are particularly powerful.

Apart from being able to use the classic hierarchical submenu structure, it is also possible to jump from one menu to another specifying its name or supplying one or more words to search for in the description of the options. This capability is available not only from within MenuFinder, but also from the OpenVMS prompt ($).

The navigation line

Records the last 10 menus called-up, allowing an extremely rapid means to recall them.

Rapid return to the last used menu

In case it is necessary to exit MenuFinder and return completely under the control of OpenVMS, it is still possible to recall immediately the last menu used.

It's simple to construct a menu

MenuFinder is an easy to use product: it is sufficient to know a few DCL commands, be able to use EDIT, the standard OpenVMS editor and know the few (10 or so) instructions for the generation of the menus. The construction of a menu is interactive and every change has an immediate effect on the screen.

Logical Names

The use of Logical Names both for the personalization of MenuFinder and for the definition of logical directories, makes the management of the product extremely flexible even in a cluster environment.

2. HOW TO BEGIN

2.1. The Tutor Menu

The best way to understand MenuFinder is to use the `TUTOR` menu. Through a series of demonstration sessions the `TUTOR` menu allows the user to gain a broad view of the capability of the product and to learn to create menus.

To call-up the `TUTOR` menu, enter the command:

```
$ MENU TUTOR
```

then:

```
CHOICE: 2
```

press the Enter key and follow the instructions given in the course of the session.

2.2. Structure Of The Guide

This guide is structured so as to offer a gradual learning process for MenuFinder and, as such, the various sections should be read in order.

Section 8 provides a complete example of the construction of a menu (and can be carried out at the computer).

The advanced user functions are described in section 9. Although not essential, their use permits the optimal management of the product.

3. HOW TO USE THE MENU

3.1. The Function Keys

The last line of the menu lists the numbers, which indicate the Function-keys used with MenuFinder. Let's see which Function-keys they correspond to and which alternative keys can be used:

Number Indicated on Last Line	**Corresponding Function Key**	**Equivalent Numeric Keypad**	**Other Equivalent Keystrokes**
KP1	F11	1	Ctrl-I
KP2	F12	2	
KP3	F13	3	LeftArrow
KP4	F14	4	RightArrow
KP5	F15	5	
KP6	F16	6	
KP7	F17	7	PageUp
KP8	F18	8	PageDown
KP9	F19	9	
KP0	F20	0	Ctrl-Z

The numeric keypad makes the activation of the function keys particularly simple when a terminal emulator on a personal computer is used.

3.2. The Meaning Of The Menu Fields

If the F11 or KP1 key is pressed and in the field

CHOICE:

there is no data, MenuFinder displays a sequence of windows which show the significance of the menu fields, together with some elementary instructions on their use.

Try it!

3.3. How The Help Key Works

Pressing the KP1 key at any time causes a text to be displayed, which indicates the possible functions and the way in which to effect them.

In particular, if in the field

CHOICE:

an option of the menu has been entered, and the corresponding menu item on the screen is indicated with a "?", then pressing F11 or KP1 key displays the associated help text.

3.4. How To Select An Option

In order to select an option, you must enter it in the field

CHOICE:

and then press the Enter or Return key.

The UpArrow and DownArrow keys may also be used to select and option.

After an option has been executed the menu from which it was requested is displayed again.

3.5. Note Of Last Option Requested

A symbol is always placed by the side of the last option requested as a note for the user.

3.6. Visualization Of A Text

If the option is of type "**t**;" it indicates that the text can be displayed by MenuFinder. The text must be in plain ASCII form, that is, without the control characters typical of documents produced with word-processor programs. The text may extend over more than one page. To understand how to move through the pages you can press the KP1 help key. By pressing the KP9 key the entire text can be printed with one of the eight proposed commands.

3.7. How To Jump Directly To A Menu

This function allows the user to call-up a menu simply by specifying a logical name or the name of the relative MDF (see the section THE MDF).

For example, to call-up the menu GAMES it is sufficient to press the KP5 key, type in the field:

Menu Name: *GAMES*

and the press the Enter key.

It is also possible to jump directly to this menu from the OpenVMS prompt by typing the command:

$ *MENU GAMES*

3.8. How To Search For A Word In The Menus

MenuFinder offers a search function which displays the menus according to the phrases present in the options window. Using the KP6 function you can make a request to MenuFinder of the type:

"Show me the menu that has, in the description of an option, the word CIRCULAR."

To obtain this result, you must first press the KP6 key then write in the field:

Word to search: *CIRCULAR*

and press the Enter key.

MenuFinder will then propose all the menus in which the word CIRCULAR is found. When the desired menu is displayed confirm it with the KP9 key and the menu can then be used. Also, the first option of the menu that satisfied the search is placed in the CHOICE field, ready for immediate selection with the Enter key. You can also ask MenuFinder to search for the options in whose descriptions appear either one or both, of two specified words. If, for example, you wanted to find the options in which both the word COMPUTER and the word INVOICING are present, you can enter in the field:

Word to search: *COMP AND INVOIC*

If, however, you wanted to find an option in which either the word POST or the word MAIL is present in the description, you could enter in the field:

Word to search: *POST OR MAIL*

3.9. How To Enter DCL Commands

By pressing the KP9 key it is possible to enter any DCL command or request the execution of any program or DCL procedure.

3.10. Special Keys

By pressing the Ctrl key and the B key together, the last eight commands or menu options entered will be displayed.

Pressing the Ctrl and W keys simultaneously causes the entire screen to be re-displayed. This function may be useful to refresh the display if, for example, messages have overwritten the screen with the REPLY command.

3.11. How To Exit MenuFinder

To exit MenuFinder you can press the F20 key or the KP0 key on the numeric keypad.

3.12. How To Copy The Screen To A Queue

By pressing the Ctrl and V keys simultaneously a copy of the screen image is sent to a print queue. The name of the print queue can be personalised (see the paragraph "The Logical Name `MENU$SCREENQUE`").

3.13. How To Copy The Screen To A File

By pressing the Ctrl and F keys simultaneously the screen image is copied into a file. This, for example allows the user, to include copies of the menu in the documentation. The name of the file can be personalised (see the paragraph "The Logical Name `MENU$SCREENFILE`").

4. THE MDF

This section describes how a menu is defined. Knowledge of the concepts presented here is necessary for the creation or modification of menus.

4.1. What Is A MDF

MDF is the acronym for Menu Description File.

A menu is described in a standard text file in which are inserted some simple instructions. MenuFinder is capable of directly interpreting this text and of displaying the described menu on the screen.

The instructions can be written in either uppercase or lowercase and the syntax is very simple.

4.2. The Description Of A Menu

The following details the MDF `USER.MEN` which generates the menu `USER`, as used in the demo sessions.

This MDF is used as a reference for the description of all the instructions.

```
NAME=USER

TITLE=+Foreign Relations Office
TITLE=

SCREEN=
! Example of comment with the "!" character

OPTION=1
DESCRIPTION=General Utilities
MENU=DUA0:[DEMO]GENERAL.MEN

SCREEN=
```

```
OPTION=2
DESCRIPTION=Notes on work to be done
TEXT=DUA0:[DEMO]USERTODO.TXT

SCREEN=

OPTION=3
DESCRIPTION=Fax received during the day
PROCEDURE=DUA0:[DEMO]FAX.COM
HELP=DUA0:[DEMO]FAX.HEL

SCREEN=

OPTION=4
DESCRIPTION=Documents Editor
COMMAND=EDITDOC

OPTION=6
DESCRIPTION=Games
MENU=DUA0:[DEMO]GAMES.MEN
SCREEN
SCREEN=Contact Mr. Pasquale 4589 for information
```

4.3. The NAME Instruction

```
NAME=USER
```

This line specifies the name to appear on the first line of the menu and in the navigation line. The name must not be longer than 7 characters (if it is longer the name is truncated).

RULE: The instruction `NAME`, if used, must be the first instruction in the MDF.

Note: The name indicated by this instruction is only for descriptive purposes in the menu and can only be used to access the menu via the "Jump to Menu" function if `USER` is defined as a logical name.

4.4. The TITLE Instruction

```
TITLE=+Foreign Relations Office
TITLE=
```

The titles specified by these instructions appear on the second and subsequent screen lines. The + sign (after the = sign) indicates that the text is to be centre justified on the line.

The second instruction generates a blank line on the screen. Up to four title lines can be defined in this way.

RULE: The `TITLE` instruction if used, must follow the `NAME` instruction. If the `NAME` instruction is not present, "`TITLE`", must be the first instruction in the MDF.

4.5. The SCREEN Instruction

With this instruction MenuFinder is requested to write a phrase in the window where the options appear.

In the following line, no phrase is specified, however, this does not cause a blank line to be displayed.

```
SCREEN=
```

With the following line, MenuFinder is requested to display the specified phrase:

```
SCREEN= Contact Mr. Pasquale 4589 for information
```

The position of the `SCREEN` instruction determines on which line the phrase is displayed.

The instruction can also be used to display on subsequent lines the continuation of a long option description, or to display a title for a group of options in the menu.

The phrases specified by the `SCREEN` instruction are also searched during the execution of the "Search Word" function to find a menu.

4.6. The OPTION Instruction

The name of an option may be up to 8 characters (alphanumeric).

4.7. The DESCRIPTION Instruction

This is used to associate a description with an option.

4.8. Groups Of Instructions For The Options

To display an option on the menu it is necessary to use a group of instructions to define the name of the option, the description and the option type.

In each group the first instruction must be `OPTION`.

4.9. The TEXT Instruction

```
OPTION=2
DESCRIPTION=Notes on work to be done
TEXT=DUA0:[DEMO]USERTODO.TXT
```

To associate a text with an option it is sufficient to specify the text file name using the `TEXT` instruction in a group of instructions.

4.10. The PROCEDURE Instruction

```
OPTION=3
DESCRIPTION=Fax received during the day
PROCEDURE=DUA:[DEMO]FAX.COM
```

To associate a procedure with an option it is sufficient to specify the file name using the `PROCEDURE` instruction in a group of instructions. Parameters can also be specified after the file name.

4.11. The HELP Instruction

```
OPTION=3
DESCRIPTION=Fax received during the day
PROCEDURE=DUA:[DEMO]FAX.COM
HELP=DUA0:[DEMO]FAX.HEL
```

This instruction associates with an option (in this case option 3) a file that contains help text.

The instruction causes the "?" character to be displayed to the right of the option, indicating to the user the presence of a help text.

The text can be called-up using the KP1 key (after the option has been entered in the “CHOICE” field).

Help text can be associated with any option type.

4.12. The COMMAND Instruction

```
OPTION=4
DESCRIPTION=Documents Editor
COMMAND=EDITDOC
```

Unlike the `PROCEDURE` instruction, where the command or program name must be embedded in a batch file, the `COMMAND` instruction allows direct specification of the command or program in the MDF.

Parameters can also be specified with command ; in the following example `DUA0:[DEMO]MANUAL.DOC` is the parameter:

```
COMMAND=EDITDOC  DUA0:[DEMO]MANUAL.DOC
```

4.13. Blank Lines In The MDF

A blank line in the MDF has no effect on the generated menu. It can be used simply to separate instruction groups, and therefore, to improve legibility of the MDF.

4.14. Comments In MDF

```
! Example of comment with the "!" character
```

A line that begins with an exclamation mark is ignored by MenuFinder and therefore has no effect on the menu. It can be used by the author of the MDF to insert notes and comments.

4.15. Abbreviated Forms Of The Instructions

Instead of the full instruction names, it is possible to use abbreviated forms, as follows:

Complete Name	Abbreviated Form
SCREEN=	S=
OPTION=	O=
DESCRIPTION=	D=
MENU=	M=
TEXT=	T=
PROCEDURE=	P=
COMMAND=	C=
NAME=	N=
TITLE=	I=
HELP=	H= or ?=
EXIT=	E=

The abbreviated forms can be interpreted by all national versions of MenuFinder.

5. THE MENUFINDER COMMANDS

The MenuFinder commands allow the user & administrator to create, modify, delete and print MDF files, text files and batch files directly, without having to manually specify their names.

Three types of command are defined:

;E	to create or modify a file
;P	to print a file
;D	to delete a file

The command must always be preceded by the semicolon ";" character.

The commands may be relative to the file associated with an option (e.g. DCL procedure) or to the MDF of the menu currently displayed.

5.1. The Edit Command (;E)

If the menu USER is currently being displayed, then:

CHOICE: *2;E*	Activates the editor for the text file associated with option 2.
CHOICE: *3;E*	Activates the editor for the batch file associated with option 3.
CHOICE: *3;E?*	Activates the editor for the help text file associated with option 3.
CHOICE: *;E*	Activates the editor for the MDF of the menu currently displayed (on exit from the editor, the menu is immediately updated to reflect any changes made).

5.2. The Print Command (;P)

If the menu USER is currently being displayed, then:

CHOICE: *2;P* Prints the text associated with option 2.

CHOICE: *3;P?* Prints the help text file associated with option 3.

CHOICE: *;P* Prints the MDF of the current menu.

5.3. The Delete Command (;D)

If the menu USER is currently being displayed, then:

CHOICE: *2;D* Deletes, after requesting confirmation, the text associated with option 2.

CHOICE: *3;D?* Deletes, after requesting confirmation, the help text file associated with option 3.

CHOICE: *;D* Deletes, after requesting confirmation, the MDF of the current menu.

The editor, the type of deletion and all the print commands can be personalised (see the following section).

6. PERSONALIZATION OF THE LOGICAL NAMES

The personalization of MenuFinder by the user is achieved through the file `SYS$LOGIN:LOGIN.DMF.`

This file is a standard DCL procedure, where the values of the logical names for MenuFinder management are defined.

6.1. What is A Logical Name

A logical name can represent the value of a parameter, the name of a file, the name of a directory or the name of an MDF.

The syntax for defining a logical name is the following:

$ *DEFINE logical_name value_represented*

For example, to associate the file `DUA0:[MENU]TEST.MEN` with the logical name `MENU$DEFAULT`, you should use the definition:

$ *DEFINE MENU$DEFAULT DUA0:[MENU]TEST.MEN*

If the value represented is expressed by more than one word, then it must be enclosed in double-quotes:

$ *DEFINE MENU$EDIT "EDIT/EDT "*

Understanding and using logical names is very important for the correct use of MenuFinder. The use of logical names by the user who creates menus is not obligatory but can, however, bring many benefits in terms of simplifying management.

6.2. The File SYS$LOGIN:LOGIN.DMF

The personal customisation of MenuFinder is achieved using the configuration file `LOGIN.DMF` in a user's login directory.

`LOGIN.DMF` is a standard DCL command file, where values for a series of logical names can be defined. The user can modify this file to insert logical names of personal menus, to define the root of his/her menu tree, to change the editor, etc.

The `SYS$LOGIN:LOGIN.DMF` file is not essential for MenuFinder to function; in its absence the logical names defined by the MenuFinder administrator remain active.

In the succeeding paragraphs, we'll discuss the configuration logical names and the menu logical names, using the following LOGIN.DMF file as an example:

```
$ DEFINE      MENU$DEFAULT         DUA0:[DEMO]USER.MEN
$ DEFINE      MENU$TOP             DUA0:[DEMO]USER.MEN
$ DEFINE      MENU$EDIT            "EDIT/EDT #"
$ DEFINE      MENU$DELETE          "DEL #"
$ ! Print Commands
$ DEFINE      MENU$PRINT_C1        "PRINT #"
$ DEFINE      MENU$PRINT_D1        "PRINT Command"
$ DEFINE      MENU$PRINT_C2        "PRINT /QUE=LN03 #"
$ DEFINE      MENU$PRINT_D2        "PRINT Command on LN03"
$ ! Menu Logical names
$ DEFINE      USER                 DUA0:[DEMO]USER.MEN
$ DEFINE      GAMES                DUA0:[DEMO]GAMES.MEN
```

6.3. The Logical Name MENU$DEFAULT

```
$ DEFINE      MENU$DEFAULT  DUA0:[DEMO]USER.MEN
```

The definition of this logical name indicates to MenuFinder which menu should be displayed, if the user does not specify any name when launching MenuFinder.

With the logical name defined as above, if the user enters the command:

$ *MENU*

the menu described in the file DUA0:[DEMO]USER.MEN will be displayed.

During the installation of MenuFinder, a predefined MDF is set for MENU$DEFAULT in order to allow the users to call-up an initial menu.

6.4. The Logical Name MENU$TOP

```
$ DEFINE        MENU$TOP              DUA0:[DEMO]USER.MEN
```

The logical name MENU$TOP is defined in order to indicate to MenuFinder the root of a menu structure.

The definition of the root is essential for use of the "Search word" function to find a menu. MenuFinder initiates its word search from the root and continues through all the menus of the tree structure hanging off the root menu.

Standalone menus can be created, which are not inserted in the tree structure, however, the "Search word" function will ignore them.

6.5. The Logical Name MENU$EDIT

```
$ DEFINE       MENU$EDIT             "EDIT/EDT   #"
```

With this logical name, the DCL command is defined which is used by MenuFinder to activate the editor following the command ;E.

Before executing the command indicated, MenuFinder substitutes the # symbol with the name of the file.

If the # symbol is not present, the file name is appended at the end of the command.

6.6. The Logical Name MENU$DELETE

```
$ DEFINE       MENU$DELETE   "DEL   #"
```

This line defines for MenuFinder the DCL command to generate when a file is to be cancelled with the ;D command. Before executing the command, MenuFinder always asks for confirmation.

The same considerations apply to the # symbol as indicated for the logical name MENU$EDIT.

6.7. The Logical Name MENU$PRINT

```
$ ! Print Commands
$ DEFINE      MENU$PRINT_C1        "PRINT #"
$ DEFINE      MENU$PRINT_D1        "PRINT Command"
$ DEFINE      MENU$PRINT_C2        "PRINT /QUE=LN03 #"
$ DEFINE      MENU$PRINT_D2        "PRINT Command on LN03"
```

Whenever MenuFinder is requested to print something (whether by the KP9 key while displaying a text, or by the ;P command in the "CHOICE" field), a list of predefined commands is displayed on the screen. A particular print option is chosen by entering a number between 1 and 8.

To add a command to the list it is necessary to define a pair of logical names.

In the above example, to display the first line in the list

```
1 -     PRINT Command
2 -     PRINT Command on LN03
Choose a number: 1
```

the description is defined by `MENU$PRINT_D1` (`D` stands for description and `1` is the position in the list) and the command is defined by `MENU$PRINT_C1` (`C` stands for command and 1 is the position in the list).

The same considerations apply to the # symbol as indicated for the logical name `MENU$EDIT`.

During installation a list is predefined that contains only the `PRINT` command to the system queue.

6.8. Logical Names Of The Menus

```
$ ! Menu Logical names
$ DEFINE      USER          DUA0:[DEMO]USER.MEN
$ DEFINE      GAMES         DUA0:[DEMO]GAMES.MEN
```

Without these logical names directly calling a menu from OpenVMS or with the function "Jump to menu" could be problematic since the full name of the MDF would have to be specified.

For example, to call-up the menu `USER` from the OpenVMS prompt requires the command:

$ *MENU DUA0:[DEMO]USER.MEN*

The presence in SYS$LOGIN:LOGIN.DMF of the line:

```
$ DEFINE       USER    DUA0:[DEMO]USER.MEN
```

allows the user the simpler command option:

$ *MENU USER*

6.9. Logical Names of a Directory

It is also possible to define logical names for a directory. The use of such logical names makes administration and travel between menus more flexible as it eliminates the necessity to always remember the full directory path and name as they are held internally in the MDF, Advantages of this facility are for example, it is possible to substitute all of the relative files of a menu in another directory or in another disk, thus to swop all of the files in the directory DUA0:[DEMO] for the directory A$DUA3:[TEST]. Normally, this would require that you substitute all of the strings DUA0:[DEMO] with A$DUA3:[TEST], however, with MenuFinder it is sufficient simply to modify the LOGIN.DMF file as follows, simply change the line,

$DEFINE A$DEMO DUA0:[DEMO]

to

$DEFINE A$DEMO A$DUA3:[TEST]

and the problem is solved without modifying the MDF.

6.10. Reserved Logical Names

Logical names which begin with `MENU$` are reserved by MenuFinder. To avoid conflict it is thus advisable to refrain from creating new logical names with this prefix.

6.11. How To Modify The File LOGIN.DMF

To modify one's own LOGIN.DMF it is sufficient to give the command:

$ *EDIT SYS$LOGIN:LOGIN.DMF*

To enable the modifications, it is then necessary to execute the file with the command:

$ *@SYS$LOGIN:LOGIN.DMF*

7. HOW TO CALL-UP MENUFINDER

7.1. Calling-up A Menu

With the command `MENU`, one can request a display of the default menu ($ *MENU*) or a specific menu ($ *MENU USER*)

7.2. Calling-up The Last Used Menu

The command `LMENU` (Last Menu) calls-up the menu that was being displayed the last time MenuFinder was exited.

The command to give is simply:

$ *LMENU*

The menu is redisplayed as it was left before exiting.

7.3. Searching For A Menu

The search for a menu is activated when one or two words are searched for in the option descriptions or in the phrases defined with the `SCREEN` instruction.

The word search can be activated not only from within the program with the KP6 key, but also from the OpenVMS prompt with the `FMENU` command Find Menu).

For example, with the command

$ *FMENU TEST*

MenuFinder is requested, to find all the menus in which the word `TEST` appears.

Abbreviated words can also be specified, for example:

$ *FMENU TES*

In this case, the menus identified are the same as those found with TEST, plus also all the others in which the character sequence TES appears within a word. The command:

$ *FMENU TEST CONNEC*

requests that all the menus are identified in which, in at least one line, the sequence

TEST CONNEC appears (intended as two words, separated by a space). It is also possible to request a search using the logical operators AND and/or OR.

For example, the command:

$ *FMENU PRO A FLOWER (where "A" stands for AND)*

requests all menus to be identified in which, in the same line, are present both the word PRO and the word FLOWER; in this case AND is interpreted as a logical operator and not as a word to be searched for.

However, the command

$ *FMENU PRO O FLOWER (where "O" stands for OR)*

requests all menus to be identified in which, in at least one line, appears either the word PRO or the word FLOWER. Commands of the following type are also possible:

$ *FMENU COMMAND LIST O FLOWER*

which, requests a search for the menu in which, in at least one line, appears the sequence COMMAND LIST or the word FLOWER.

7.4. Displaying A Text

The command SFILE (Show FILE) allows a text to be displayed on the screen just as if the request had been made from within a menu (instruction TEXT=) and it is therefore possible to request the printing of the text in the same way.

The command to give, is simply:

$ *SFILE file_name*

7.5. Creating A Menu

To create a menu from the OpenVMS prompt, MenuFinder provides the command `CMENU` (Create MENU).

Normally menus are created and modified from within MenuFinder, but to create the first menu (or to create a completely new menu tree) `CMENU` may be useful.

The syntax of the command is:

```
$ CMENU file_name
```

where “file_name”, is intended as the name of the MDF that is to be created.

With this command MenuFinder creates the MDF file indicated and inserts example lines to facilitate the definition of the menu.

In the following sections, all the phases for creating a menu are considered, starting right from the command `CMENU`.

8. STEPS IN CREATING A MENU STRUCTURE

8.1. Objectives

We propose to recreate a part of the menu structure seen in the demo session.

Therefore, we must:

- Create the MDF of a menu `USER`;
- Define `USER` as a logical name;
- Define the menu `USER` as the root node of the tree structure;
- Define the menu `USER` as the menu that should be displayed when the `MENU` command is entered from OpenVMS without specifying anything else.

These objectives allow us to follow all phases that are normally performed to create a menu structure.

The steps described can be executed directly at the terminal.

8.2. How To Create The First Menu

First of all, we create the sub-directory in which all the files of the menu structure will be placed. Therefore we enter the following OpenVMS command:

```
$ CREATE/DIRECTORY DUA0:[DEMO]
```

```
$ CMENU DUA0:[DEMO]USER.MEN
```

The `CMENU` command copies to the description file `DUA0:[DEMO]USER.MEN` a reference MDF and displays the menu so obtained.

Now, the `MDF DUA0:[DEMO]USER.MEN` must be modified.

The modifications can be made with the MenuFinder commands:

CHOICE:*;E*

This command activates the editor defined by the logical name MENU$EDITOR.

The contents of the file DUA0:[DEMO]USER.MEN are displayed on the screen as follows:

```
NAME=Example
TITLE=
TITLE=+Sample of centered title (with + character)
TITLE=
! This line is a comment
! Message line on the screen
SCREEN=Sample of message line
! Blank line on the screen

SCREEN=

OPTION=1
DESCRIPTION=Description of the DCL procedure option
PROCEDURE=sys$login:sample.com
HELP=sys$login:sample.hel

! Blank line on the screen
SCREEN=

OPTION=2
DESCRIPTION=Description of the text option
TEXT=sys$login:sample.txt

SCREEN=

OPTION=3
DESCRIPTION=Description of the menu option
MENU=sys$login:sample.men
[EOB]
```

Modifications must be made in order to obtain the following file:

```
NAME=USER

TITLE=+Foreign Relations Office
TITLE=
SCREEN=


OPTION=1
DESCRIPTION=General Utilities
MENU=DUA0:[DEMO]GENERAL.MEN

SCREEN=

OPTION=2
DESCRIPTION=Notes on work to be done
TEXT=DUA0:[DEMO]USERTODO.TXT

SCREEN=

OPTION=3
DESCRIPTION=Fax received during the day
PROCEDURE=DUA:[DEMO]FAX.COM
HELP=DUA0:[DEMO]FAX.HEL

SCREEN=

OPTION=4
DESCRIPTION=Documents Editor
COMMAND=EDITDOC

OPTION=6
DESCRIPTION=Games
MENU=DUA0:[DEMO]GAMES.MEN
SCREEN
SCREEN=Contact Mr. Pasquale 4589 for information
```

Once the text has been modified and the editor exited the updated menu will immediately be displayed on the screen.

Let's continue with our construction of the menu.

8.3. How To Create A Submenu

To create the submenu relative to option 6 it is necessary to activate once again the editor, specifying that you intend to work with the file associated with option 6. The command to give is:

CHOICE: *6;E*

MenuFinder identifies the name of the file to use with the editor by analysing in the current MDF the group of instructions relative to option 6, and determines that the file name is `DUA0:[DEMO]GAMES.MEN`

The editor displays the empty file and it is therefore necessary to add the following instructions:

```
NAME=Games
TITLE=%
SCREEN=
SCREEN=              GAMES

OPTION=1
DESCRIPTION=Tetris
PROCEDURE=DUA0:[DEMO]GAME1.BAT

OPTION=2
DESCRIPTION=Chess
PROCEDURE=DUA0:[DEMO]GAME2.BAT

OPTION=3
DESCRIPTION=Battleships
PROCEDURE=DUA0:[DEMO]GAME3.BAT

OPTION=4
DESCRIPTION=The cat on the wall
PROCEDURE=DUA0:[DEMO]GAME4.BAT
```

When the text has been completed you can exit the editor.

Now that option 6 has been defined it can be used. Enter:

CHOICE: *6*

a press the Enter key.

The GAMES menu is displayed as described above.

To be able to call-up the option of the GAMES menu you should now transfer to the DUA0:[DEMO] directory the batch files that contain the relative game programs.

Obviously it is also possible to call-up the games directly from the directory where they are installed (it is sufficient to put in the MDF the complete paths of the respective command files).

Note also that, if the game can be run simply by entering the program name, it is not necessary to use a command file: it is sufficient to change the instruction PROCEDURE to COMMAND and insert in this instruction the name of the program prefixed with the directory name.

Note that we have created a submenu without leaving MenuFinder.

In the same way, we can create all the other submenus.

8.4. How To Create A Text

Let's return now to the menu USER to prepare option 2.

To achieve this just press the LeftArrow key (to select the menu to the left of the navigation line) and press Enter.

You must now again call the editor. Enter:

CHOICE: *2;E*

The editor displays the empty file DUA0:[DEMO]TODO.TXT. We must enter the following text:

```
Monday 12th
*  Close the annual activity plan
*  Telephone RICTEL to confirm quotation

Tuesday 13th
Prepare report on new initiatives

Wednesday 14th
Doctors appointment at 16:00

Thursday 26th
Accompany visitors
```

```
Friday 27th
Prepare for move on Monday

Monday 30th
Move
```

having left the editor, the text just entered can be displayed with the command:

CHOICE: *2*

8.5. How To Create A Batch File

The batch file associated with option 3 is created in the same way as a text file. The command to give in this case, however, is:

CHOICE: *3;E*

In the empty file that is displayed, you must insert the necessary commands to call up the hypothetical program that displays faxes.

8.6. How To Create A Help Text

The help text associated with option 3 is created in the same way as a text file. The command to give in this case, however, is:

CHOICE: *3;E?*

In the empty file that is displayed, you must insert the necessary instructions on how to use the hypothetical program for displaying the faxes.

At the end, to display the help text, just enter:

CHOICE: *3*

and then press the KP1 key.

8.7. Define A Logical Name For A Menu

We must now define the name `USER` such that the menu just created can be called directly from the OpenVMS prompt with the command:

$ *MENU USER*

and from MenuFinder with the KP5 "Jump to menu" key:

Name of menu: *USER*

To define the logical name `USER`, we must modify the file `LOGIN.DMF` with the command:

$ *EDIT SYS$LOGIN:LOGIN.DMF*

to insert the line:

```
$ DEFINE USER DUA0:[DEMO]USER.MEN
```

To enable the logical name just inserted we must give the command:

$ *@SYS$LOGIN:LOGIN.DMF*

To verify all is as it should be, enter the command:

$ *MENU USER*

The menu `USER` will be displayed.

8.8. Define The Root Of A Structure

We define the menu `USER` as the root of the menu structure in order to be able to use the "Search word" function. We must again modify the `LOGIN.DMF` file with the command:

$ *EDIT SYS$LOGIN:LOGIN.DMF*

to insert the line:

```
$ DEFINE MENU$TOP DUAO:[DEMO]USER.MEN
```

To enable the logical name just inserted we must give the command:

```
$ @SYS$LOGIN:LOGIN.DMF
```

To verify all is as it should be, enter the command:

```
$ FMENU TETRIS
```

MenuFinder proposes as a result the menu with the game `TETRIS`. It is sufficient to confirm the menu with the KP9 key and then press Enter to run the game.

8.9. Define The Default Menu

All that remains now is to define the menu `USER` as the default menu, that is the menu that should be displayed when the command `MENU` - without parameters - is given from the OpenVMS prompt.

We must once again modify the file `LOGIN.DMF` with the command:

```
$ EDIT SYS$LOGIN:LOGIN.DMF
```

to insert the line:

```
$ DEFINE MENU$DEFAULT DUA0:[DEMO]USER.MEN
```

To enable the logical name just inserted we must give the command:

```
$ @SYS$LOGIN:LOGIN.DMF
```

To verify all is as it should be, enter the command:

```
$ MENU
```

MenuFinder will display the menu `USER`.

9. ADVANCED USER FUNCTIONS

9.1. The COMMAND Instruction With The # Variable

We have seen in section 4 how it is possible to pass a parameter to a program or to a command with an instruction of the form:

```
COMMAND=EDITDOC DUA0:[DEMO]MANUAL.DOC
```

The file `DUA0:[DEMO]MANUAL.DOC` is however seen by MenuFinder as a simple parameter to pass to the batch file `EDITDOC` and, as such, is not able to perform any operation on this file such as activation of the editor, printing, copying, etc.

The way in which to insert the file parameter amongst those which MenuFinder can manage is to insert the filename in the instruction # as shown in the example:

```
#=DUA0:[DEMO]MANUAL.DOC
COMMAND=EDITDOC #
```

With this instruction, MenuFinder completes the `EDITDOC` command substituting the # symbol with the file name.

The # symbol in the second line is essential only if in the command other parameters are inserted after the file name.

In case no other parameters are present and the # symbol is missing, MenuFinder always adds the value of # to the end of the command line.

Utilizing this pair of instructions, the type of option that appears in a menu changes from "C" to "#" to denote that the file name used as parameter can be used by MenuFinder.

9.2. The Global Symbol MFPAUSE

When MenuFinder is activated, the global symbol MFPAUSE is defined, this is useful if the user wishes to pause at the end of a command before returning to the menu.

The syntax to utilise the `MFPAUSE` symbol is:

```
COMMAND=MFPAUSE par1 par2 ... par8
```

If, for example, the following instruction is used:

```
COMMAND=MFPAUSE SHOW TIME
```

at the end of the command `SHOW TIME`, MenuFinder requests the user to press the Return key, before it returns to the menu.

9.3. Composition Of DCL Commands

We have seen that with the KP9 function key, MenuFinder makes it possible to execute any program, DCL command or batch file.

This functionality is very powerful as it allows direct insertion in the command line of the names of the files present in the MDF.

This allows, for example, copying of the text file associated with a menu option to the file associated with an option of another menu; or it is possible to use the cut and paste functions on options between menus (see the `EMENU` command in the following paragraph).

Some simple examples will serve as an illustration (see the `TUTOR` menu for an interactive demonstration of this functionality):

Suppose we want to see how many bytes the MDF file of the currently displayed menu occupies.

Pressing KP9 the command line is shown:

$

Type in the line the command `DIR/SIZE` followed by a space. Then press the KP6key.

In the command line we thus obtain:

```
$ DIR/SIZE DUA0:[MENU]USER.MEN
```

Pressing the UpArrow or DownArrow keys to select the various options, the description of the KP6 key changes according to the option selected.

In particular, if the description of KP6 is:

M	File	the name of the file is that of the current MDF being displayed on the screen;
t	File	the name of the file is that of the text file associated with the option;
p	File	the name of the file is that of the batch-file associated with the option;
m	File	the name of the file is that of the MDF of the sub-menu associated with the option;
#	File	the name of the file is that of the variable associated with the option (see the previous section);

No description is displayed, however, when the option type is a simple command and the # instruction is not used.

In case a help text file is also associated with the option, then the description of the function key KP5 is "? File". If you want to insert the name of the help text file in the DCL command line, then you must press KP5.

While constructing the command line the KP5 and KP6 keys can be pressed as many times as required.

For example, to construct the COPY command of a text file associated with an option to another text associated with a different option, you could move to the first option and press the KP6 key; then move to the second option and press again KP6.

While constructing a command, you can also change menu, utilising the navigation keys LeftArrow and RightArrow. This way it is possible, for example, to copy a file associated with an option in one menu to the file associated with an option in a different menu.

9.4. Cut And Paste Of Options Between Menus

When a menu structure begins to get complex, it may become necessary to reorganise it. For example, some options of a menu could be regrouped as a submenu. To copy or move options between menus the EMENU command is useful. This command requires as parameters the names of the two MDF's of the menus involved.

In practice EMENU activates the editor VAX EVE on the two files: the cut and paste functions are actually those of this editor. To activate EMENU with the names of the two MDF's the composition functions for DCL commands detailed in the previous section can be utilised.

This way it is very simple to construct a command of the form:

$ *EMENU MYMENU:BACKUP.MEN MENU$MINE:SAVEALL.MEN*

and perform the cut and paste operations between the two buffers containing the Menu Description Files.

For those not familiar with the EVE editor, note that to activate one of the buffers of the list proposed by EMENU, just select it with the up or down arrow and then press Return.

To redisplay the buffer list, press the DO key and then enter the command:

Command: *SHOW BUFFER*

9.5. The EXIT Instruction

Inserting the EXIT instruction in a group in which a COMMAND or PROCEDURE instruction is present, requests MenuFinder to return to the OpenVMS environment after execution of the command or batch file associated with the option. For example, with the following instruction group:

```
OPTION=3
DESCRIPTION=Read Faxes received today
PROCEDURE=DUA0:[DEMO]FAX.COM
HELP=DUA0:[DEMO]FAX.HEL
EXIT=Y
```

after having chosen option 3, at the end of execution of the command file, the menu is not redisplayed, instead you are returned to the OpenVMS environment.

9.6. The Logical Name MENU$SCREENQUE

This logical name allows the user to personalise the name of the print queue utilised for printing the screen image following a CTRL-V. For example:

```
$ DEFINE MENU$SCREENQUE "LN03$QUEUE"
```

If this logical name is not defined, the print queue utilised is `SYS$PRINT`.

9.7. The Logical Name MENU$SCREENFILE

This logical name allows the user to personalise the name of the file to which the screen image is copied following a CTRL-F. For example:

```
$ DEFINE MENU$SCREENFILE "OUTPUT.TMP"
```

If this logical name is not defined, the file name utilised is `SYS$LOGIN:MENU$SCREEN.LIS`.

9.8. The Logical Name MENU$BORDER_M

Using this logical name it is possible to eliminate the menu window side borders. In the case of terminals which function with only a low line speed, this is useful to speed up the display of a menu.

For example:

```
$ DEFINE MENU$BORDER_M  NO
```

If this logical name is not defined or if the associated value is "YES", the menu window side borders will be displayed.

If this characteristic is required only for certain menus it is sufficient to insert in the corresponding MDF, before any instructions, the line:

```
BORDER_M=N
```

The value specified in this line (whether Y or N) overrides the value of the logical name `MENU$BORDER_M`, if any.

9.9. The Logical Name MENU$BORDER_T

Using this logical name it is possible to eliminate text window side borders (instruction TEXT=). In the case of terminals which function with only a low line speed, this is useful to speed up the display of a menu.

For example:

```
$ DEFINE MENU$BORDER_T  NO
```

If this logical name is not defined or if the associated value is "YES", the text window side borders will be displayed.

If this characteristic is required only for certain menus it is sufficient to insert in the corresponding MDF, before any instructions, the line:

```
BORDER_T=N
```

The value specified in this line (whether Y or N) overrides the value of the logical name MENU$BORDER_T, if any.

9.10. The Logical Name MENU$OPT_TYPE

Using this logical name it is possible to eliminate from the menus the option types (c, p, m, #, t) and replace them with the character "-".

Example:

```
$ DEFINE MENU$OPT_TYPE NO
```

If this logical name is not defined or if the associated value is "YES", the option types will be displayed.

If this characteristic is required only for certain menus it is sufficient to insert in the corresponding MDF, before any instructions, the line:

```
OPT_TYPE=N
```

The value specified in this line (whether Y or N) overrides the value of the logical name MENU$OPT_TYPE, if any.

9.11. The Logical Name MENU$OPT_RIGHT

Utilising this logical name it is possible to left justify, instead of right justify, the option name in all menus.

This definition permits the user, for example, to structure more clearly, in a menu, the paragraphs of an on-line manual.

Example:

```
1     Introduction to the computer centre
1.1      Working hours
1.2      People to contact
2     Available products
2.1      Programming languages
2.2      CAD products
```

To define this logical name, use the command:

```
$ DEFINE MENU$OPT_RIGHT NO
```

If this logical name is not defined or if the associated value is "YES", right justification occurs.

If this characteristic is required only for certain menus it is sufficient to insert in the corresponding MDF, before any instructions, the line:

```
OPT_RIGHT=N
```

The value specified in this line (whether Y or N) overrides the value of the logical name `MENU$OPT_RIGHT`, if any.

9.12. Reading Global Variables

To report internally to the Descriptive File of a menu the value of a global symbol it is sufficient to initialise the symbol, for example in the LOGIN.COM, with the command,

$ flag = = " "

and insert {flag} internally in the MDF in the correct position as desired.

For example, it could be used to create a check list for a Menu,

OPTION=1

DESCRIPTION={flag} execute JOB1

COMMAND=@JOB.COM ;flag = = "!"

The execution of option 1 is now achieved with the special customised symbol "!".

N.B. the character ";" semicolon is used to concatenate several commands together. It must be preceded by at least one space.

other examples of the use of global symbols are present in the menu PUBLIC.