OS9 - System Utilities

S-Screen Control

M-Menuing system

P-Point and Shoot File Selection

4072 E. 22nd St.
Suite 178
Tucson, Arizona 85711
1 (602) 745-2327

M - Menuing system

Commands update. September 2, 1989

This file contains an update to the printed documentation.

M rev 11 1234 bytes

New control line \x disables the x exit key. When this line is present in a menu file the menu cannot be exited.

When M is run from a startup file starting m the ex command will reduce the memory overhead by 8k be sure to redirect both input and output when runing m from a script file. i.e.

ex m<>>>/1 not m</1 this needs to be done because m modifies the window descriptor.

M edition history:

****

August 24, 1989	version 1.0 (10 by ident) first commercial release.
August 28, 1989	version 1.0 bug fixed in highly commented menus.
September 2, 1989	version 1.1 break bug in parameter passing fixed \x exit override control line added.

************ ************************** *************************

M - R-MENU SYSTEM

by Rick R. Roth

4072 E. 22nd St.
Suite 178
Tucson, Arizona 85711
1 (602) 745-2327

M - R-MENU SYSTEM

INTRODUCTION

M is a small machine language version of R-Menu a menu system written for Unix based systems in C. M is designed to merge in with shell and become a memory resident menuing system. M was purposely written in machine language rather than C to enable its use on floppy based systems as well as systems equipped with hard drives. (The Unix version written in C using the curses windowing package compiles to a little over 50k this version written for OS9 Level 2 running on the color computer is 1218 bytes long!)

The actual menus used by M are ASCII files created by your favorite word processor.
The format for the menu files are quite simple and the actual syntax is identical to your standard command line. The command line syntax requires that some lines end with a space followed by carriage return. (You must be sure that your editor will allow you to do this. Edit will, others will not.)

M

Syntax: M [menu filename]

Function: A quick and easy to use memory resident menu system, M (or R-Menu) displays a menu centered on the screen in an overlay window. Displaying the menu in an overlay window means that the screen is preserved as it appeared when M was called, this can be changed from within the menu to clear the screen before the menu is displayed. The menu size is dependent on the size of the menu file called and the number of items displayed. There is no waste space nor are empty selections displayed.

M defaults to a black border, a blue background, white foreground, and the current menu selection highlighted in red. The current selection is also pointed to by an arrow -=> on the left margin. The copyright appears on the top border and an instruction line " Press S to call Shell, Press X to Exit" appears of the bottom border in letters in the foreground color. These defaults can be changed from within the menu.

The menu is centered horizontally and vertically on a 80 column screen and centered vertically on a 40 column screen. M checks the size of the screen before displaying the menu and adjusts the size and position accordingly. This means that a program called by M can change the screen size without harming the menu display upon return to the menu.

M allows you to pass parameters to a menu command line, ie a filename to an editor or a path to the directory command.

Items are selected from the menu by either moving the arrow to the selection desired and pressing entered or if keyboard selection is enabled by pressing the key corresponding to the selection.

Menu File Structure:

A M menu file consists of four types of lines, the required Menu Title, Selection lines, Command lines, and optional Control lines.

The **Menu Title** line MUST be the FIRST non control line in a file, the Title line consists of a maximum of 33 characters ended with a carriage return. This is the line displayed at the top of your menu. ie
Rick's Main Menu<cr>

The next two types of lines MUST appear in pairs, first the **Selection line** again 33 characters followed by a <cr>, and then the **Command line** that preforms the task described in the Selection line. The command line can be ANY valid OS9 command line. If the command line will work from a script file or from the command line prompt the line will work in R-Menu. The command line is limited to a maximum of 254 characters ended with a space and a carriage return <cr>.
ie:
1. Directory of Memory<cr>
mdir <cr>

COMMAND LINE OPTIONS

R-Menu provides 3 command line options to add to the versatility of the system. The first is the **pause command.** If the first character of a command line is a minus(-) the menu will run the command and then print the message

Press any key to continue:

at the bottom of the display and then wait for a keystroke before restoring the menu display. The pause option MUST be the first character on a command line. To display a directory and then pause to give you time to read it the command line should be:

-dir <cr>

alternately if you have the public domain pause command the line could be:

dir;pause Press any key to continue <cr>

either would give the same result.

The second command line option is the **Shell bypass command.** The shell bypass is activated by a plus(+) as the first or second character in a command line. If the pause command is also used the bypass command (+) MUST be the second character if the pause is not used the bypass MUST be the first character.

The shell bypass causes M to fork to the program on the command line directly, without calling a new version of the shell to interpret the command line. The built in shell commands, <, >, >>, #, &, ;, !, and () can not be used in a command line that uses the shell bypass. For more information on the shell bypass see the HOW M WORKS section at the end of the documentation.

The following command lines use the bypass in a correct manner.

+m /d1/asm/source/util.m <cr> - calls a new menu.
-+dir <cr> - displays a directory then pauses.
-+list m.doc <cr> - lists a file and then pauses.

The third command line option is the **Parameter passing command.** This option is activated by a dollar sign($) in the first, second, or third character in a line. The parameter command must come after the pause and bypass commands if they are used but before the body of the command line. Command lines that take advantage of the parameter passing abilities are slightly different from a normal command line. The syntax for such a command line is:

$ prompt string $command $ [$...]<cr>

The prompt string may be 33 characters long and is displayed in the border of the parameter window that appears after the menu disappears but before the command line is executed. This parameter window is also an overlay window and will not effect the screen below. As M parses the command line the cursor will return to the parameter window and wait for input every time a $ is encountered on the command line after the first command. The information entered in the parameter window is then substituted for the $ in the command line before it is executed.

The following command line will prompt you to enter two file-names and then copy from the first file to the second.

```
$Enter source/destination $copy #40k $ $ <cr>
```

The parameter passed can be up to 40 characters and can be concatenated to allow entry of more then 40 characters ie.

```
$Enter note$date>+/dd/sys/log;echo $$$$ >+/dd/sys/log <cr>
```

Will allow you to add a note of up to 160 characters to a file called log if the sys directory. (The ">+" is a shell+ command and appends information to the end of a file and it is not available with the vanilla shell.)

When parameters are passed to a command line the line is expanded in a 500 byte buffer, this limits the number of parameters passed and the size of the parameters, but since most versions of shell only allow a 256 byte command line this limitation should not impose a great hardship.

More than one command line option can be present. If more than one is used the pause must come first followed by the shell bypass and then the parameter passing option.

The fourth and last type of line in the menu file is the **Control line**. There are four types of control lines, all control lines start with a backslash as the first character of the line. Control lines may appear anywhere within the file, but should not be below the twelfth Selection/command line pair in a large menu. ALL control lines are optional and do not need to be in the file for proper operation.

The control lines are:

\ - Comment lines. Comment lines begin with a backslash followed by a space and end with a carriage return. Comment lines are ignored by M and are only there your to document your menus. ie

\ This is a comment line it has no effect on the action of the menu.

\c - Color control lines. The default menu colors can be changed very easily between menus by use of the color control line. The format of a color control line is backslash c followed by four numbers and ending with a carriage return. No spaces are allowed in this line. The four numbers are the foreground, background, border, and highlight color. The colors are determined by the pallet selected and the default number/color assignment can be found on page 3-6 of the OS9 Windowing System Section of the OS9 Level II manual. The default palette is:

0	White	1	Blue	2	Black	3	Green
4	Red	5	Yellow	6	Magenta	7	Cyan

Using this information the line \c1047 will create a menu with a blue foreground, a white background, a red border, and the current menu selection highlighted in cyan letters. Remember the copyright message and the exit instructions appear on the top and bottom borders in the foreground color, if the foreground and border colors are the same the message will not appear.

\C - Clear Screen Control line. This control line causes the menu to clear the screen before displaying the menu. The menu appears in an overlay window in the screen save mode anything that was on the screen when the menu was called will remain on the screen and be visible when the menu exits. The \C overrides this.

\s - Keyboard select control line. This final control line enables the keyboard select option the format for this line is backslash s followed by the keyboard character used to select each line on the menu there must be one character in the line for each selection. ie

\s1234 pressing 1 will select the first item, 2 the second, 3 the third, and 4 the fourth.

\sabcd pressing a will select the first etc.

The only characters to be avoided in the keyboard select line are e and s which are predefined to call a new shell or exit the menu. All other keyboard characters can be used. The characters defined in the keyboard select line, s and x, the up and down arrows, and the carriage return are the only keys the program recognizes all other keys including the break key and control c sounds a beep, and the highlighted selection blinks. The keyboard select line is optional, if not included in the file the only way to select an item on the menu is with the arrow keys.

INCLUDED ON THIS DISK IN A DIRECTORY NAMED DEMOS ARE SEVERAL SAMPLE MENUS AND A WORKING DEMONSTRATION. To view a demonstration of M place the distribution disk in drive 1, cd /d1, and type start<cr>.

HOW M WORKS

When M is called the program checks the command line for a filename, if no name is present M loads a file named MENU.m from the current directory, if a name or path/name is present this file is read and loaded. This file contains a Menu Title and several Selection/Command line pairs. The pairs are loaded into a buffer and indexed by pointers, when a command is selected the next pointer points to the command line to be processed. Command lines are normally processed by the OS9 command line processor, named SHELL.

Shell normally resides in memory and is not thought of as a program, but it is much as M is. During normal operation M calls another incarnation of shell and passes your command line to it to interpret. This is the same thing that happens when you run a procedure file. Running a command line in this way lets you process a command line that contains multiple statements as well as i/o redirection, concurrent execution (&), and memory allocation (#40k). This works well but the extra incarnation of shell requires another 8K of data space as well as needing to map shell into the work space. This gives you an effective work area of 48k.

M has a VERY simple command line interpreter built into it that will allow you to process a single program and its parameters without the overhead of calling another incarnation of shell. M will not interpret any of the built in shell commands, these must be avoided when using the shell bypass option.This will save you the 8k data area reserved for the extra shell, and avoid mapping shell into your work area. This gives you a full 64k to run your program! You can observe this using pmap from another window. The shell bypass should make M a viable alternative even on a 128k machine.

M was written on a 512k Radio Shack computer with a Diskmaster disk drive. The assembly code was written with the XED editor from Microtech consultants, and compiled with the level 1 assembler ASM. The level 2 debug utility from the developers toolkit was used extensively.