NAME

ccfe menu − the Curses Command Front-end menu definition file.

DESCRIPTION

The Curses Command Front-end menu is a tool which permits to collect scripts or CCFE forms in hierarchical menus. When the user selects an item of a menu, then is executed the corresponding **action**, which can be an O.S. command, an external script, a **CCFE form** or another **CCFE menu**. It is based on the menu library, the curses extensions for programming menus: please see the **ncurses**(3X) and **menu**(3X) manpages. A CCFE menu can be a *static menu* or a *dynamic menu*: in the first case all the menu definition (both general definition and items definition) is contained in a single file, in the second case they are splitted in many files: it is a simple way to distribute software managed by CCFE (please see **DYNAMIC MENUS** below).

SYNOPSIS

A menu is definited in a file with name terminated with the *.menu* extension, and the following attributes can be used in that file:

```
title {}
top {}
path {}
item {}
item {}
 ...
item {}
```

(some of them are optional).
Comments begins with the "#" character, and continue until the end of the line. Empty lines are ignored.

GENERAL MENU ATTRIBUTES

title { STRING }

Title of the menu to display.
It is also used as *fast path description* when **ccfe(1)** is called with **-f** option.
If title attribute is omitted, it will be used the *description* of the item selected from the calling menu, but if omitted in a menu called with a *fast path* (see ccfe(1)), then no title will be used.
Optional: Yes
Examples:

```
title { Configure System Services }
```

or

```
title {
  Configure System Services
}
```

top { STRING1\nSTRING2 }

Text lines (max 2) to display on the bottom of the menu, between the title and the items of the menu. If omitted, the lines of the parameter *form_top_msg* in **ccfe.conf**(5) will be displayed.
Optional: Yes
Examples:

```
top {
  Move cursor to desired item and press Enter.
}
```

bottom { STRING1\nSTRING2 }

Not yet implemented.

path { LIST:OF:PATHNAMES }

List of extra path (added to environment variable PATH) to search binaries calling the O.S. in the menu **action{}**. Note: **path{}** is used only to search binaries, and not for forms and menus.
Optional: Yes

Formato:
- LIST:OF:PATHNAMES has the character ':' as separator;
- LIST:OF:PATHNAMES can contain relative and absolute subdirectories;
- le directory con path relativo sono relativizzate a $SCREEN_DIR

Examples:
path { ldap_lib }

NB: XX_abc_XX indica una frase da rivedere o tradurre.

ITEM DEFINITION

item {} Defines an item of the menu. It must appear at least one time in a menu definition file. Every item definition is a collection of attributes, with the following syntax:

```
item {
        attribute     = value
        attribute     = value
                       ...
        attribute     = value
}
```

Valid item attributes are explained later with the following format:

attribute_name

Attribute description.

Values: Notes on the syntax of the value of the field. If there is a list of values, the first element of the list is the default. The "" value represents the null value.
Required: Yes or *no*, depending if the attribute is mandatory or not.
Types: List of field **type** for which this attribute is valid.

id

Identifier of the item used to refer its value by help procedure. It must be unique in the scope of the menu.

Values: An upper case string (but it is case insensitive) composed by characters A-Z,0-9,-,_
Required: Yes
Types: All

descr

Text of the item to put in the menu.

Values: A string of any charachter.
Required: Yes

action

Action to execute when this item is highlighted and the **<Enter>** key is pressed. The selection of items without this attribute have the same effect to press the **<back key>**. Items without this attribute defined can be marked; please see the **mark_noaction_items** parameter in the **ccfe.conf**(5) manual page. The syntax for a menu action is the following:

action { *action-type*[**(***options***)**]**:** *action-args* **}**

Where:

action-type can be one of the following:

run Execute *action-args* and then show the results in the **Output Browser screen** (please see the **ccfe**(1) manpage). *action-args* can be a simple command or a complete script which

will be passed to a command shell for parsing (with "sh -c"). The shell used is the one specified in the **shell** parameter of the `ccfe.conf` configuration file.

menu Load and display the specified menu. Before the name of menu, you can specify a path relative to /usr/share/ccfe directory (for example `demo.d/sysmon`).

form Load and post the specified form. Note that it is not possible to pass arguments as when calling a form from another form. Before the name of form, you can specify a path relative to /usr/share/ccfe directory (for example `users.d/ask_user`).

system Temporarily exit from **ccfe**, executes *action-args* and returns in **ccfe**. This is useful for actions which need user interaction: for example the **passwd**(1) command. If **wait_key** is specified in *options*, then it is displayed a message reporting the exit status and wait a user keystroke before to return in **ccfe**.

exec Terminate **ccfe** and executes *action-args*. This is useful to return to Operating System after calling external commands or applications. Be aware that *action-args* are executed directly instead of via a command shell, so invocation errors may not be reported.

options is a comma-separated list of one or more of the following options:

confirm
Open a pop-up window to request a confirm before execute action. The option highlighted by default is "No" (not confirm).

log Save in the logfile the output of the execution of the action.

wait_key
Valid only for *action-type*=**system**: when action is completed, wait for continuing until the user presses a key.

Required: No

DYNAMIC MENUS

A *dynamic menu* is a directory whose files are used to build the menu description: the file named **definition** must exists and contains general menu attributes (like **title{}**, **top{}** and **bottom{}** blocks). Every item description is contained in a file: its name is user defined, but the extension must be **.item**. ***.item** files are sorted, so you can establish the items order by the name of their files definition. At least one **.item** file must exists.

For example, the dynamic menu *demo* is defined by the following files:

```
/usr/lib/ccfe/ccfe/demo.menu/
/usr/lib/ccfe/ccfe/demo.menu/definition
/usr/lib/ccfe/ccfe/demo.menu/recursive.item
/usr/lib/ccfe/ccfe/demo.menu/sysmon.item
```

where

- `/usr/lib/ccfe/ccfe/demo.menu/definition` content is:

```
title {
  CCFE Demo Menu
}
```

- `/usr/lib/ccfe/ccfe/demo.menu/recursive.item` content is:

```
item {
  id     = RECURSIVE
  descr  = Test form recursivity
  action = form:demo.d/recursive
}
```

- `/usr/lib/ccfe/ccfe/demo.menu/sysmon.item` content is:

```
item {
  id     = SYSMON
  descr  = System resources usage monitors
  action = menu:sysmon
}
```

Please see the samples included in the CCFE distribution package.

EXAMPLES

To be filled in!

SEE ALSO

ccfe(1), **ccfe.conf**(5), **ccfe_form**(5), **ccfe_help**(5), **curses**(3X), **menu**(3X)