SABER-C REFERENCE MANUAL

Version 1.2

Table of Contents

NAME

action – execute a debugging action

SYNOPSIS

```
void action()

void action(line)
int line ;

void action(function)
void (*func)() ;

void action(filename, line)
char *filename ;
int line ;
```

OPTIONS

`terse_list` (unset)

DESCRIPTION

`Action` allows users to execute statements at different locations in a program without having to modify the source code. It can be used to make temporary changes to a program, such as adding debugging code, `printf` statements, and consistency checks. Actions cannot be set within compiled code.

`Action` prompts the user for a single statement to execute; multiple statements can be executed by enclosing them within a block. The action may use any global, formal, or automatic variables that are in scope at the location where the action is set.

If `action` is called without an argument, the action is executed at every line in every function. The action is not performed if the user is executing statements or expressions directly in the environment or at a break level. If the argument is a line number, a function, or a file and a line number, the action will be executed at the specified location only.

Actions can be listed with the `status` command and deleted with the `delete` command.

OPTIONS

If the **terse_list** option is set, the `list` command will not display the complete text of an active action when listing the location where the action was triggered.

EXAMPLE

The example below sets an action that prints the value of `total` every time line `10` of file `main.c` is reached. If the value of `total` equals `1000`, the `stop` command is called and execution is suspended. Notice that a block is used because the action consists of more that one statement.

```
1 -> action("main.c", 10)
Setting action at "f.c", line 3, main().
Use braces for multiple statments.
Enter <Ctrl-D> to quit, <Ctrl-C> to redo.
action -> {
action +>   printf("Value = %d\n", total) ;
```