NAME

exit, return, goto – shell built-in functions to enable the execution of the shell to advance beyond its sequence of steps

SYNOPSIS

sh

exit [*n*]
return [*n*]

csh

exit [**(***expr***)**]
goto *label*

ksh

† **exit** [*n*]

† **return** [*n*]

DESCRIPTION

sh

exit will cause the calling shell or shell script to exit with the exit status specified by *n*. If *n* is omitted the exit status is that of the last command executed (an EOF will also cause the shell to exit.)

return causes a function to exit with the return value specified by *n*. If *n* is omitted, the return status is that of the last command executed.

csh

exit will cause the calling shell or shell script to exit, either with the value of the status variable or with the value specified by the expression *expr* .

The **goto** built-in uses a specified *label* as a search string amongst commands. The shell rewinds its input as much as possible and searches for a line of the form *label***:** possibly preceded by space or tab characters. Execution continues after the indicated line. It is an error to jump to a label that occurs between a **while** or **for** built-in command and its corresponding **end**.

ksh

exit will cause the calling shell or shell script to exit with the exit status specified by *n*. The value will be the least significant 8 bits of the specified status. If *n* is omitted then the exit status is that of the last command executed.When **exit** occurs when executing a trap, the last command refers to the command that executed before the trap was invoked. An end-of-file will also cause the shell to exit except for a shell which has the **ignoreeof** option (See **set** below) turned on.

return causes a shell function or **'.'** script to return to the invoking script with the return status specified by *n*. The value will be the least significant 8 bits of the specified status. If *n* is omitted then the return status is that of the last command executed. If **return** is invoked while not in a function or a **'.'** script, then it is the same as an **exit**.

On this man page, **ksh**(1) commands that are preceded by one or two † (daggers) are treated specially in the following ways:

1. Variable assignment lists preceding the command remain in effect when the command completes.
2. I/O redirections are processed after variable assignments.
3. Errors cause a script that contains them to abort.

4. Words, following a command preceded by †† that are in the format of a variable assignment, are expanded with the same rules as a variable assignment. This means that tilde substitution is performed after the = sign and word splitting and file name generation are not performed.

SEE ALSO

break(1), **csh**(1), **ksh**(1), **sh**(1)