NAME

doconfig – execute a configuration script

SYNOPSIS

cc [*flag* ...] *file* ... **–lnsl** [*library* ...]

 # include <sac.h>

int doconfig(int *fildes*, **char** **script*, **long** *rflag***);**

MT-LEVEL

Unsafe

DESCRIPTION

doconfig( ) is a Service Access Facility library function that interprets the configuration scripts contained in the files **</etc/saf/***pmtag***/_config>**, **</etc/saf/_sysconfig>**, and **</etc/saf/***pmtag* / *svctag***>**, where *pmtag* specifies the tag associated with the port monitor, and *svctag* specifies the service tag associated with a given service. See **pmadm**(1M) and **sacadm**(1M).

script is the name of the configuration script; *fildes* is a file descriptor that designates the stream to which stream manipulation operations are to be applied; *rflag* is a bitmask that indicates the mode in which *script* is to be interpreted. If *rflag* is zero, all commands in the configuration script are eligible to be interpreted. If *rflag* has the **NOASSIGN** bit set, the **assign** command is considered illegal and will generate an error return. If *rflag* has the **NORUN** bit set, the **run** and **runwait** commands are considered illegal and will generate error returns.

The configuration language in which *script* is written consists of a sequence of commands, each of which is interpreted separately. The following reserved keywords are defined: **assign**, **push**, **pop**, **runwait**, and **run**. The comment character is **#**; when a **#** occurs on a line, everything from that point to the end of the line is ignored. Blank lines are not significant. No line in a command script may exceed 1024 characters.

assign *variable=value*
: Used to define environment variables. *variable* is the name of the environment variable and *value* is the value to be assigned to it. The value assigned must be a string constant; no form of parameter substitution is available. *value* may be quoted. The quoting rules are those used by the shell for defining environment variables. **assign** will fail if space cannot be allocated for the new variable or if any part of the specification is invalid.

push *module1*[, *module2*, *module3*, . . .]
: Used to push STREAMS modules onto the stream designated by *fildes*. *module1* is the name of the first module to be pushed, *module2* is the name of the second module to be pushed, etc. The command will fail if any of the named modules cannot be pushed. If a module cannot be pushed, the subsequent modules on the same command line will be ignored and modules that have already been pushed will be popped.

pop [*module*]
: Used to pop STREAMS modules off the designated stream. If **pop** is invoked with no arguments, the top module on the stream is popped. If an argument is given,

modules will be popped one at a time until the named module is at the top of the stream. If the named module is not on the designated stream, the stream is left as it was and the command fails. If *module* is the special keyword ALL, then all modules on the stream will be popped. Note that only modules above the topmost driver are affected.

runwait *command*
: The **runwait** command runs a command and waits for it to complete. *command* is the pathname of the command to be run. The command is run with **/usr/bin/sh –c** prepended to it; shell scripts may thus be executed from configuration scripts. The **runwait** command will fail if *command* cannot be found or cannot be executed, or if *command* exits with a non-zero status.

run *command*
: The **run** command is identical to **runwait** except that it does not wait for *command* to complete. *command* is the pathname of the command to be run. **run** will not fail unless it is unable to create a child process to execute the command.

Although they are syntactically indistinguishable, some of the commands available to **run** and **runwait** are interpreter built-in commands. Interpreter built-ins are used when it is necessary to alter the state of a process within the context of that process. The **doconfig( )** interpreter built-in commands are similar to the shell special commands and, like these, they do not spawn another process for execution. See **sh**(1). The built-in commands are:

cd
ulimit
umask

RETURN VALUES

doconfig( ) returns **0** if the script was interpreted successfully. If a command in the script fails, the interpretation of the script ceases at that point and a positive number is returned; this number indicates which line in the script failed. If a system error occurs, a value of **–1** is returned. When a script fails, the process whose environment was being established should *not* be started.

SEE ALSO

sh(1), **pmadm**(1M), **sacadm**(1M)

NOTES

This interface is unsafe in multithreaded applications. Unsafe interfaces should be called only from the main thread.