NAME

nohup – run a command immune to hangups

SYNOPSIS

/usr/bin/nohup *command* [*arguments*]
/usr/xpg4/bin/nohup *command* [*arguments*]

AVAILABILITY

/usr/bin/nohup SUNWcsu

/usr/xpg4/bin/nohup SUNWxcu4

DESCRIPTION

The **nohup** utility invokes the named *command* with the arguments supplied. When the *command* is invoked, **nohup** arranges for the **SIGHUP** signal to be ignored by the process.

nohup can be used when it is known that *command* will take a long time to run and the user wants to logout of the terminal; when a shell exits, the system sends its children **SIGHUP** signals, which by default cause them to be killed. All stopped, running, and background jobs will ignore **SIGHUP** and continue running, if their invocation is preceded by the **nohup** command or if the process programmatically has chosen to ignore **SIGHUP**.

/usr/bin/nohup Processes run by **/usr/bin/nohup** are immune to **SIGHUP** (hangup) and **SIGQUIT** (quit) signals.

/usr/xpg4/bin/nohup Processes run by **/usr/xpg4/bin/nohup** are immune to **SIGHUP**.

nohup does not arrange to make processes immune to a **SIGTERM** (terminate) signal, so unless they arrange to be immune to **SIGTERM** or the shell makes them immune to **SIGTERM**, they will receive it.

If **nohup.out** is not writable in the current directory, output is redirected to **$HOME/nohup.out**. If a file is created, the file will have read and write permission (**600**, see **chmod**(1)). If the standard error is a terminal, it is redirected to the standard output, otherwise it is not redirected. The priority of the process run by **nohup** is not altered.

OPERANDS

The following operands are supported:

command The name of a command that is to be invoked. If the *command* operand names any of the special **shell_builtins**(1) utilities, the results are undefined.

arguments Any string to be supplied as an argument when invoking the *command* operand.

EXAMPLES

It is frequently desirable to apply **nohup** to pipelines or lists of commands. This can be done only by placing pipelines and command lists in a single file, called a shell script. One can then issue:

example$ nohup sh *file*

and the **nohup** applies to everything in *file*. If the shell script *file* is to be executed often, then the need to type **sh** can be eliminated by giving *file* execute permission.

Add an ampersand and the contents of *file* are run in the background with interrupts also ignored (see **sh**(1)):

example$ nohup *file* **&**

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **nohup**: **LC_CTYPE**, **LC_MESSAGES**, **PATH**, and **NLSPATH**.

HOME Determine the path name of the user's home directory: if the output file **nohup.out** cannot be created in the current directory, the **nohup** command will use the directory named by **HOME** to create the file.

EXIT STATUS

The following exit values are returned:

126 *command* was found but could not be invoked.

127 An error occurred in **nohup**, or *command* could not be found

Otherwise, the exit values of **nohup** will be that of the *command* operand.

FILES

nohup.out the output file of the **nohup** execution if standard output is a terminal and if the current directory is writable.

$HOME/nohup.out the output file of the **nohup** execution if standard output is a terminal and if the current directory is not writable.

SEE ALSO

batch(1), **chmod**(1), **csh**(1), **ksh**(1), **nice**(1), **sh**(1), **shell_builtins**(1), **signal**(3C), **environ**(5)

WARNINGS

If you are running the Korn shell (**ksh**(1)) as your login shell, and have **nohup**'ed jobs running when you attempt to logout, you will be warned with the message

You have jobs running.

You will then need to logout a second time to actually logout; however, your background jobs will continue to run.

NOTES

The C-shell (**csh**(1)) has a built-in command **nohup** that provides immunity from **SIGHUP**, but does not redirect output to **nohup.out**. Commands executed with '**&**' are automatically immune to **HUP** signals while in the background.

nohup does not recognize command sequences. In the case of the following command

example$ nohup command1; command2

nohup applies only to **command1**. The command

example$ nohup (command1; command2)

is syntactically incorrect.