The exec builtin command

Synopsis

```
exec [-a NAME] [-cl] [COMMAND] [ARG...] [REDIRECTION...]
```

Description

The `exec` builtin command is used to

- **replace** the shell with a given program (executing it, **not as new process**)
- set redirections for the program to execute or for the current shell

If only redirections are given, the redirections affect the current shell without executing any program.

Options

Option	Description
`-a NAME`	Passes `NAME` as zeroth argument for the program to be executed
`-c`	Execute the program with an empty (cleared) environment
`-l`	Prepends a dash (`-`) to the zeroth argument of the program to be executed, similar to what the `login` program does

Exit status

- on redirection errors it returns 1, otherwise 0
- on exec failures:
  - a non-interactive shell terminates; if the shell option execfail is set `exec` returns failure
  - in an interactive shell, `exec` returns failure

Examples

Wrapper around a program

```
myprog=/bin/ls
echo "This is the wrapper script, it will exec $myprog"

# do some vodoo here, probably change the arguments etc.
# well, stuff a wrapper is there for

exec "$myprog" "$@"
```

Open a file as input for the script

```
# open it
exec 3< input.txt

# for example: read one line from the file(-descriptor)
read -u 3 LINE
# or
read LINE <&3

# finally, close it
exec 3<&-
```

Overall script logfile

To redirect the whole `stdout` and `stderr` of the shell or shellscript to a file, you can use the `exec` builtin command:

```
exec >/var/adm/my.log 2>&1

# script continues here...
```

Portability considerations

- POSIX® specifies error code ranges:
  - if `exec` can't find the program to execute, the error code shall be 126
  - on a redirection error, the error code shall be between 1 and 125
- the `-a NAME` option appeared in Bash 4.2-alpha
- POSIX® does **not** specify any options for `exec` (like `-c`, `-l`, `-a NAME`).

See also

- Redirection

Discussion

Vitaliy (http://void.net.ua/wiki/bash:start), 2012/07/25 07:36 ()

I would add here Bash "diamond" operator for opening files for reading and writing:

```
exec <>file
```

Jan Schampera, 2012/08/12 07:02 ()

Agreed, it should be crosslinked to the redirections section.

I never heard "diamond operator" - nice

Peter Green, 2014/06/20 13:27 ()

I came to this page while trying to work out how to redirect the output of the current script to both a file and the console.

My first attempt was

exec 2>&1 | tee output.txt

But this didn't work, asking on irc I was told on irc (by pgas)this was because "exec is executed in a subshell" and "each side of a | run in forked subshells"

A suggestion given on irc (by pgas) was

exec > >(tee output.txt) 2>&1

This sort of worked but had the undesirable side affect that the final output of the script was written to the terminal after the script finished rather than the script waiting.

The final soloution I ended up with (also suggested by pgas)was to wrap the main body of my script in a function and redirect the output of the function.

main() {

```
  #do stuff here.
```

}

main "$@" 2>&1 | tee output.txt

Bash Hackers Wiki