NAME
  command — pass a command to the shell and return stdout and stderr

SYNOPSIS
  #include <command.h>

  int
  command(const char *string, char *outbuf, int outlen, char *errbuf,
           int errlen);

DESCRIPTION
  The command() function hands the argument string to the command interpreter sh(1). Any output generated by the command on stdout is placed into the buffer outbuf; any output generated on stderr is placed into the buffer errbuf. In either case, command() will only write up to outlen and errlen bytes to the respective buffers.

  If the number of bytes written into each buffer is less than the provided length, then command will NUL-terminate the data it wrote; otherwise, the buffer is not terminated.

  The calling process waits for the shell to finish executing the command, ignoring SIGINT and SIGQUIT, and blocking SIGCHLD.

  If string is a NULL pointer, command() will return non-zero, if the command interpreter is available, or zero if none is available. Otherwise, command() returns the termination status of the shell in the format specified by waitpid(2).

RETURN VALUES
  If a child process cannot be created, or the termination status of the shell cannot be obtained, command() returns -1 and sets errno to indicate the error. If execution of the shell fails, command() returns the termination status for a program that terminates with a call of exit(127).

SEE ALSO
  sh(1), dup2(2), execve(2), pipe(2), waitpid(2), popen(3), shquote(3), system(3)

/hdr/src/lib/libc/stdlib/system.c

HISTORY
  The command() function was first used as an in-class exercise for the class CS631 Advanced Programming in the UNIX Environment at Stevens Institute of Technology in the Fall of 2018.