17.8 Reading Passphrases

When reading in a passphrase, it is desirable to avoid displaying it on the screen, to help keep it secret. The following function handles this in a convenient way.

Function: *char* * **getpass** *(const char *prompt)*

Preliminary: | MT-Unsafe term | AS-Unsafe heap lock corrupt | AC-Unsafe term lock corrupt | See POSIX Safety Concepts.

`getpass` outputs *prompt*, then reads a string in from the terminal without echoing it. It tries to connect to the real terminal, `/dev/tty`, if possible, to encourage users not to put plaintext passphrases in files; otherwise, it uses `stdin` and `stderr`. `getpass` also disables the INTR, QUIT, and SUSP characters on the terminal using the `ISIG` terminal attribute (see Local Modes). The terminal is flushed before and after `getpass`, so that characters of a mistyped passphrase are not accidentally visible.

In other C libraries, `getpass` may only return the first `PASS_MAX` bytes of a passphrase. The GNU C Library has no limit, so `PASS_MAX` is undefined.

The prototype for this function is in `unistd.h`. `PASS_MAX` would be defined in `limits.h`.

This precise set of operations may not suit all possible situations. In this case, it is recommended that users write their own `getpass` substitute. For instance, a very simple substitute is as follows:

```
#include <termios.h>
#include <stdio.h>

ssize_t
my_getpass (char **lineptr, size_t *n, FILE *stream)
{
  struct termios old, new;
  int nread;

  /* Turn echoing off and fail if we can't. */
  if (tcgetattr (fileno (stream), &old) != 0)
    return -1;
  new = old;
  new.c_lflag &= ~ECHO;
  if (tcsetattr (fileno (stream), TCSAFLUSH, &new) != 0)
    return -1;

  /* Read the passphrase */
  nread = getline (lineptr, n, stream);

  /* Restore terminal. */
  (void) tcsetattr (fileno (stream), TCSAFLUSH, &old);

  return nread;
}
```

The substitute takes the same parameters as `getline` (see Line-Oriented Input); the user must print any prompt desired.