```
    printf("Installing handler for signal %d\n", SIGINT);
    orig_handler = signal(SIGINT, handler);
    raise_sig();

    printf("Changing handler to SIG_IGN\n");
    signal(SIGINT, SIG_IGN);
    raise_sig();

    printf("Restoring original handler\n");
    signal(SIGINT, orig_handler);
    raise_sig();

    printf("Program terminates normally\n");
    return 0;
}

void handler(int sig)
{
    printf("Handler called for signal %d\n", sig);
}

void raise_sig(void)
{
    raise(SIGINT);
}
```

Incidentally, the call of `raise` doesn't need to be in a separate function. I defined `raise_sig` simply to make a point: regardless of where a signal is raised—whether it's in `main` or in some other function—it will be caught by the most recently installed handler for that signal.

The output of this program can vary somewhat. Here's one possibility:

```
Installing handler for signal 2
Handler called for signal 2
Changing handler to SIG_IGN
Restoring original handler
```

From this output, we see that our implementation defines `SIGINT` to be 2 and that the original handler for `SIGINT` must have been `SIG_DFL`. (If it had been `SIG_IGN`, we'd also see the message `Program terminates normally`.) Finally, we observe that `SIG_DFL` caused the program to terminate without displaying an error message.

24.4 The <setjmp.h> Header: Nonlocal Jumps

```
int setjmp(jmp_buf env);
void longjmp(jmp_buf env, int val);
```