Here's an idea of how Vim reads input and transitions between modes. The actual code is much more complex but follows this basic flow:

```c
while (1) {
  normal_mode_process_byte(getc());
}

void normal_mode_process_byte(char c) {
  if (c == 'i') // enter insert mode
    while (1) {
      insert_mode_process_byte(getc());
    }
}
```

What needs to be done is remove all those read loops and replace by a function (e.g., `process_byte(char c)`) that will buffer input, delegate to mode-specific input-handling functions or transition editor state when necessary. Something like this:

```c
void process_byte(char c) {
  if (mode == NORMAL)
    normal_mode_process_byte(c);
  else if (mode == INSERT)
    insert_mode_process_byte(c);
  ...
}

void normal_mode_process_byte(char c) {
  if (c == 'i') {
    mode = INSERT;
    return;
  }
  ...
}
```

Think of the editor as a state machine: when input is read and fed into the machine, it may change the active buffer, transition mode, etc. The point is: the machine shouldn't call IO functions directly