UNIVERSITY OF CALIFORNIA AT BERKELEY
College of Engineering
Department of Electrical Engineering and Computer Sciences
Computer Science Division

CS 162
Fall 1988

John Ousterhout

Mini-Sermon #4: Programming and Documentation Style (Discipline and Craftsmanship)

- This sermon takes 20 minutes.

- Question is: how to build large software systems in a reasonable amount of time, and make them work reliably?

- Books and courses on SE tend to discuss life cycles, PERT charts, staffing, and other technical approaches. I'm going to talk more about philosophical details.

- Programming is a craft: something like a high-tech form of silversmithing. To make big systems work, it takes a tremendous amount of discipline and structure. Unfortunately, Americans aren't too good at this stuff. Fortunately, if you are, you'll have a tremendous advantage.

- Rule #0: simplicity. It's easy to make things complicated, harder to make them simple. Don't accept complexity.

- Rule #1: adopt a consistent style and use it everywhere. Decide on file organization, procedure structuring, naming conventions, location of curly braces, everything. Talk about our style manuals, impor-

tance of uniformity. By doing things consistently, they don't get omitted. Give example of documentation.

- Rule #2: don't litter. Temptation is to make fast fixes that dirty things up. It's crucial to take the time to fix things right at the beginning. You'll never have time to come back to it later!

- Rule #3: document carefully as you go. Don't put this off! Most important things are interfaces: procedure headers and data structures and other things that tie together the parts of the system. Tell about writing procedure headers for a whole module before writing the procedures.

- Rule #4: documentation should describe things at a higher level than code (but not *too* high a level). Examples of pathological cases:
  - ''Add one to i.''
  - ''Now we have it the way we want it.''

- Rule #5: Put documentation near the code: otherwise you forget to change the documentation when you change the code.

- Rule #6: Quality, not quantity. Don't need enormous amounts of documentation if what you have is high-quality. E.g. don't need 400-character variable names.

- Summary: discipline, craft. Take time up front to save time later.