# UNIT 11

## A DAY IN THE LIFE OF A COMPUTER OPERATOR /PROGRAMMER

Programmers write the code that tells computers what to do. System code tells a computer how to interact with its hardware; applications code tells a computer how to accomplish a specific task, such as word processing or spreadsheet calculating. Systems programmers must be familiar with hardware specifications, design, memory management, and structure, while application programmers must know standard user interface protocols, data structure, program architecture, and response speed. Most programmers specialize in one of the two areas.

At the start of a project, applications programmers meet with the designers, artists, and financiers in order to understand the expected scope and capabilities of the intended final product. Next, they map out a strategy for the program, finding the most potentially difficult features and working out ways to avoid troublesome patches. Programmers present different methods to the producer of the project, who chooses one direction. Then the programmer writes the code. The final stages of the project are marked by intense, isolated coding and extensive error-checking and testing for quality control. The programmer is expected to address all issues that arise during this testing.

Systems programmers may be hired on a Monday, handed the technical specifications to a piece of hardware, then told to write an interface, or a patch, or some small, discrete project that takes only a few hours. Then on Tuesday, they might be moved to a different project, working on code inherited from previous projects. Systems programmers must prove themselves technically fluent: "If you can't code, get off the keyboard and make room for someone who can," wrote one.

Both arenas accommodate a wide range of work styles, but communication skills, technical expertise, and the ability to work with others are important in general. Programmers work together respectfully; they help each other when they want to. But there are no significant professional organizations that might turn this group of people into a community.

The best features of this profession are the creative outlet it provides for curious and technical minds, the pay, which can skyrocket if a product you coded is a major success, and the continuing education. A few programmers indicated that an aesthetic sensibility emerges at the highest levels of the profession, saying that "Reading good code is like reading a well-written book. You're left with wonder and admiration for the person who wrote it."

## Paying Your Dues

Academic requirements are gaining importance for entry-level positions in the field of programming. Coursework should include basic and advanced programming, some technical computer science courses, and some logic or systems architecture classes. The complexity of what first-time programmers are asked to code is growing, as is the variety of applications, such as compatibility to the Internet and the ability to translate into a marketable CD-ROM. Long hours and a variety of programming languages--PERL, FORTRAN, COBOL, C, C++--can make the initial programmer's life a whirlwind of numbers, terms, and variables, so those who are not comfortable working in many modes at once may find it difficult to complete tasks. The programmer must remain detail-oriented in this maelstrom of acronyms. For mobility within the field, programmers should concentrate on developing a portfolio of working programs that show competence, style, and ability.

## Present and Future

Originally, in the 1960s, all software was known as "freeware," and was distributed among the few technical eggheads who built their own computers. During this period, one young Harvard student sent a letter to his fellow programmers saying that he thought freeware was a destructive concept, that people should trademark and copyright all their programs in order to receive the value of what these programs would eventually be worth. Most scoffed at this young impudent for his arrogant vision of the future of the personal computer and his denial of the 1960s sentiment of sharing and community. Young Bill Gates decided to stick to his opinions, and, $35 billion later, it is hard to argue with his success. Programming right now is understaffed and the field will continue to grow for the next five years at a fast pace. Many industries are just now realizing the benefit of having tailored and modular code written to address their specific needs. Staffing levels at many major programming corporations are expected to increase, and individual freelance "hackers for hire" are expected to become valuable outsourcing resources for these companies.

## Quality of Life

### *Present and future*

Two-year professionals work under the supervision of established programmers, handling sections of code or modular pieces of programs. Little responsibility is offered to new hires in terms of defining program architecture and creating new methods of handling data or graphics. They are, however, given reasonable autonomy on their own section of code. Satisfaction is high.

Pay is average. Many work for between one and two years at a given firm, then move to another with greater challenges.

***Five years out***

Salaries rise, and hours increase significantly. Duties include defining programming architecture, coding, and debugging more junior programmers' codes. Many have contact with executives and clients. Those who progress possess strong communication skills and understand what the client wants and needs. Travel may be a feature of the five-year programmer's life, going on-site to address client needs. A few begin their own businesses.

***Ten years out***

Ten-year professionals have either begun their own businesses as independent programmers or consolidated their positions as experienced programmers at large concerns. Hours increase, but the position becomes one more of defining program architecture, working with staffs of programmers, and managing a variety of projects. Few ten-year professionals do basic coding; a number of those who love it do some but delegate the detail work to less-experienced junior programmers. Satisfaction is high; salaries can become significant.