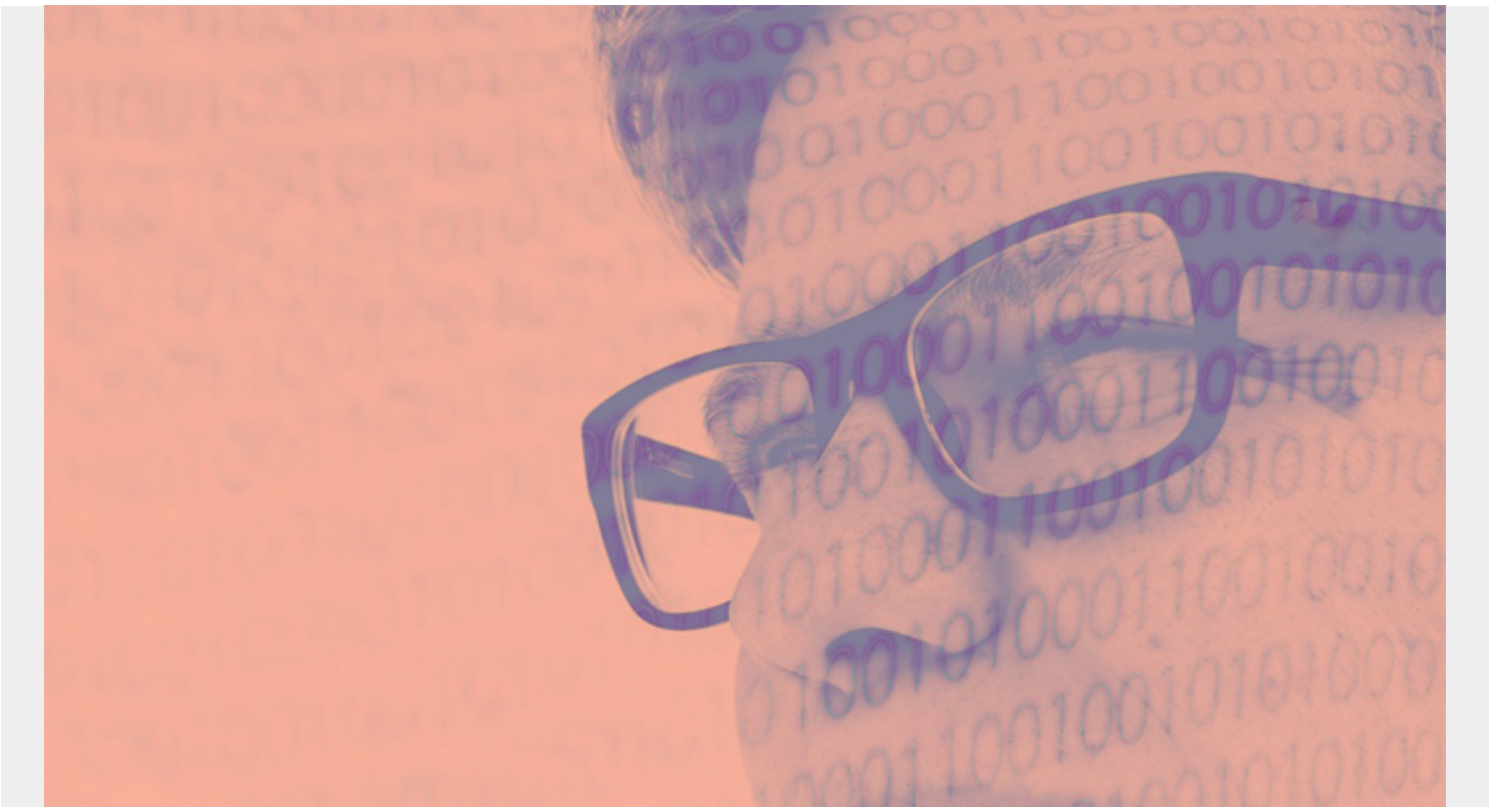
WHAT IS SYSTEMS PROGRAMMING?



If you're visiting this page, then it's safe to assume you have some appreciation for how complex the technology sector is. IT is a complicated beast in part due to the fact that it's constantly evolving and each new advancement is an added layer of complexity on top of an already impossible number of pre-existing layers.

We employ BMC Blogs as a means of peeling back some of these layers to offer deep insights and brief overviews alike—ideally exposing the IT world in a way that allows newcomers and veterans to understand it. Today, we thought we'd talk about a more basic concept in a way that clears up at least some of the confusion surrounding software development.

The topic for today, as I'm sure you guessed by reading the title of this post, is systems programming. But before we talk about what systems programming is, we should first address what a system even is within this context.

What is a System?

The dictionary definition of a system is "a set of things working together as parts of a mechanism or an interconnecting network." This is a pretty apt way of thinking about systems as they pertain specifically to the IT world. A computer system is a collection of components (both hardware and software) that function as a part of a whole.

A system is comprised of five primary elements: architecture, modules, components, interfaces, and data:

Architecture

Architecture is the conceptual model that defines the system structure and behavior. This is often represented graphically through the use of flowcharts that illustrate how the processes work and how each component is related to one another.

Modules

Modules are pieces (hardware or software) of a system that handle specific tasks within it. Each module has a defined role that details exactly what its purpose is.

Components

Components are groups of modules that perform related functions. Components are like micro-systems within the system at large. Using components and modules in this way is called modular design, and it's what allows systems to reuse certain pieces or have them removed and replaced without crippling the system. Each component can function on its own and can be interchanged or placed into new systems.

Interfaces

Interfaces encompass two separate entities: user interfaces and system interfaces. User interface (UI) design defines the way information is presented to users and how they interact with the system. System interface design deals with how the components interact with one another and with other systems.

Data

Data is the information used and outputted by the system. System designers dictate what data is pertinent for each component within the system and decide how it will be handled.

Each component complements the system in its own way to keep everything functioning properly. If one piece of the puzzle becomes askew, the entire system can be impacted. Because technology is constantly evolving, components are modified, added, or removed on a constant basis. To make sure these modifications have the desired effect, systems design is used to orchestrate the whole affair.

What is Systems Design?

Systems design involves defining each element of a system and how each component fits into the system architecture. System designers focus on top-level concepts for how each component should be incorporated into the final product. They accomplish this primarily through the use of Unified Modeling Language (UML) and flow charts that give a graphical overview of how each component is linked within the system.

Systems design has three main areas: architectural design, logical design, and physical design. The architectural design deals with the structure of the system and how each component behaves within it. Logical design deals with abstract representations of data flow (inputs and outputs within the system).

Physical design deals with the actual inputs and outputs. The physical design establishes specific requirements on the components of the system, such as input requirements, output requirements, storage requirements, processing requirements, and system backup and recovery protocols. Another way of expressing this is to say that physical design deals with user interface design, data design, and process design.

Systems designers operate within key design principles when they are creating the system architecture. Some of the key tenets of good system design are:

- Be Explicit - All assumptions should be expressed.
- Design for Iteration - Good design takes time, and nothing is ever perfect the first time.
- Keep Digging - Complex systems fail for complex reasons.
- Be Open - Comments and feedback will improve the system.

The systems programmers are the ones responsible for executing on the vision of the system designers.

What is Systems Programming?

Systems programming involves the development of the individual pieces of software that allow the entire system to function as a single unit. Systems programming involves many layers such as the operating system (OS), firmware, and the development environment.

In more recent years, the lines between systems programming and software programming have blurred. One of the core areas that differentiates a systems programmer from a software programmer is that systems programmers deal with the management of system resources. Software programmers operate within the constraints placed upon them by the system programmers.

This distinction holds value because systems programming deals with “low-level” programming. Systems programming works more closely with computer resources and machine languages whereas software programming is primarily interested in user interactions. Both types of programming are ultimately attempting to provide users with the best possible experience, but systems programmers focus on delivering a better experience by reducing load times or improving efficiency of operations.

It’s imperative that everyone working within the system is aligned. The primary goal of any service or product is to deliver value to your customers. Whether you are involved with top-level user interactions or low-level system infrastructure, the end goal remains the same. This is why a company culture that supports teamwork and goal-alignment is so important for technology companies.

Modern customers have increasingly high expectations. As such, organizations must constantly be seeking ways to improve their output to provide customers with an ever-improving product. Achieving this is done through intelligent systems design and an agile approach to development. Bringing everyone together to work towards a singular goal is the main pursuit of the DevOps approach to software development.