Kernel symbols are labels or names that represent functions, variables, and other entities within the Linux kernel's code. They serve as human-readable identifiers for specific memory addresses or code locations.

A kernel symbol table is a data structure, typically found in `/proc/kallsyms` (or `/proc/ksyms` in older kernels), that maps these symbol names to their corresponding memory addresses. This table is crucial for various aspects of kernel operation and development:

- **Debugging and Analysis:**

  The symbol table allows developers and tools to resolve symbol names to addresses during debugging sessions, crash dump analysis, and performance profiling, making it easier to understand kernel behavior and identify issues.

- **Dynamic Loading of Modules:**

  When a loadable kernel module is inserted, it often needs to access functions or data structures within the core kernel. The symbol table enables the module loader to resolve these references, linking the module to the appropriate kernel symbols.

- **Symbol Versioning:**

  The kernel uses symbol versioning to manage changes in exported symbols over time, ensuring backward compatibility for modules built against older kernel versions. This information is also part of the symbol table.

- **Security Controls:**
  The visibility of kernel symbols can be controlled through kernel configuration options (e.g., `CONFIG_KALLSYMS`, `CONFIG_KALLSYMS_ALL`) and runtime settings (e.g., `/proc/sys/kernel/kptr_restrict`) to enhance security by limiting access to kernel internals.

In essence, kernel symbols and the symbol table provide a vital mechanism for understanding, debugging, and extending the Linux kernel.