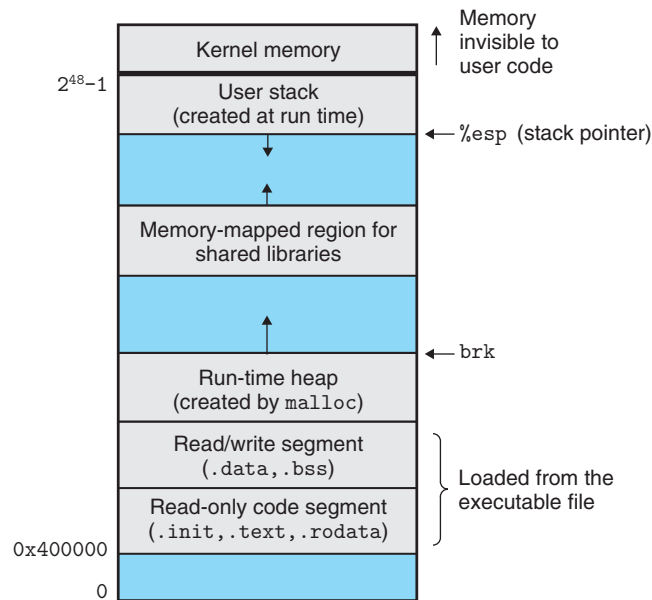


Figure 7.15

Linux x86-64 run-time memory image. Gaps due to segment alignment requirements and address-space layout randomization (ASLR) are not shown. Not to scale.



executable object file into the code and data segments. Next, the loader jumps to the program's entry point, which is always the address of the `_start` function. This function is defined in the system object file `crt1.o` and is the same for all C programs. The `_start` function calls the *system startup function*, `__libc_start_main`, which is defined in `libc.so`. It initializes the execution environment, calls the user-level `main` function, handles its return value, and if necessary returns control to the kernel.

7.10 Dynamic Linking with Shared Libraries

The static libraries that we studied in Section 7.6.2 address many of the issues associated with making large collections of related functions available to application programs. However, static libraries still have some significant disadvantages. Static libraries, like all software, need to be maintained and updated periodically. If application programmers want to use the most recent version of a library, they must somehow become aware that the library has changed and then explicitly relink their programs against the updated library.

Another issue is that almost every C program uses standard I/O functions such as `printf` and `scanf`. At run time, the code for these functions is duplicated in the text segment of each running process. On a typical system that is running hundreds of processes, this can be a significant waste of scarce memory system resources. (An interesting property of memory is that it is *always* a scarce resource, regardless