To understand the overlay concepts and practice how to overlay the current process to new process in Linux using C.

In Linux, the term "overlay" refers to the process of replacing the current process image with a new process image. This can be done using the execve() function, which overlays the current process with a new process image specified by the pathname and argument list arguments.

C program that demonstrates how to overlay the current process with a new process image:

```
#include <stdio.h>
#include <unistd.h>

int main()
{
   // Create an array of pointers to strings to be passed as arguments to the new process
   char *args[] = { "/bin/ls", "-l", NULL };

   // Overlay the current process with the "ls" command
   execve("/bin/ls", args, NULL);

   // This line will not be executed if execve() is successful
   printf("Error executing execve()\n");

   return 0;
```

}

In this example, the execve() function overlays the current process with the ls command, which lists the contents of the current directory in long format. The args array specifies the arguments to be passed to the ls command, and the third argument is a NULL pointer to indicate the end of the argument list.

The execve() function returns only if an error occurs. If the execve() function is successful, the current process is replaced by the new process image and the original process is no longer running.

It's worth noting that the execve() function is just one of the several exec*() functions available in Linux for overlaying processes. Other functions include execl(), execle(), execlp(), execv(), execvp(), and execvpe(). These functions differ in the way they accept the pathname and argument list, as well as the environment for the new process.