Problem 1:

Write a C/C++ program that redirects the output of a command to a specified file using the dup2 system call.

1- Your program should take two command-line arguments:
   a) The first argument is the command to execute (e.g., ls, cat, echo, etc.).
   b) The second argument is the name of the output file to which the standard output (stdout) of the command should be redirected.

Example Usage:

```
./redirect_command ls output.txt
```

2- Use fork, execvp, and dup2 system calls to:
   a) Create a child process.
   b) Execute the command specified in the first argument using execvp.
   c) Redirect the standard output (stdout) of the command to the file specified in the second argument using dup2.

Problem 2:

Create a file named "data.txt" in the same directory as your program. Populate this file with some text content (e.g., "Hello, Memory Mapping!").

Write a C program that performs the following tasks:

- Opens the "data.txt" file for reading and writing.
- Uses the mmap system call to map the contents of "data.txt" into memory.
- Displays the content of the mapped memory region on the console.
- Modifies the content of the mapped memory region to append " - Updated!" to the existing text.
- Displays the updated content of the mapped memory region on the console.
- Unmaps the memory region and closes the file.

Memory Maps

Memory mapping in operating systems refers to the technique of mapping a portion of a process's virtual address space directly to a physical location, which could be a file, device, or shared memory region.

Map the file into memory using *mmap()* with parameters specifying the file descriptor, size, access permissions (*PROT_READ* for read-only), mapping type (*MAP_PRIVATE* for a private copy), and offset (0 for the beginning of the file).

Unmap the memory using munmap() to release resources when done.

Example Programs:

```
#include <sys/mman.h>
#include <fcntl.h>
#include <unistd.h>


int main() {
   int fd = open("file.txt", O_RDONLY);
   off_t file_size = lseek(fd, 0, SEEK_END);
   void* mapped_data = mmap(NULL, file_size, PROT_READ, MAP_PRIVATE, fd, 0);
   // Use mapped_data as if it's a pointer to the file's contents in memory
   munmap(mapped_data, file_size); // Unmap memory when done
   close(fd);
   return 0;
}
```