```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

#define SHM_SIZE 1024

int main() {
   HANDLE hMapFile;
   LPCTSTR pBuf;

   // Create a file mapping object
   hMapFile = CreateFileMapping(
      INVALID_HANDLE_VALUE,    // Use the page file
      NULL,                    // Default security
      PAGE_READWRITE,          // Read/write access
      0,                       // High-order DWORD of the maximum size of the file mapping object
      SHM_SIZE,                // Low-order DWORD of the maximum size of the file mapping
object
      TEXT("Local\\MyFileMappingObject") // Name of the mapping object
   );

   if (hMapFile == NULL) {
      fprintf(stderr, "Could not create file mapping object (%d).\n", GetLastError());
      return 1;
   }

   // Map the file view
   pBuf = (LPTSTR)MapViewOfFile(
      hMapFile,          // Handle to the mapping object
      FILE_MAP_ALL_ACCESS, // Read/write access
      0,
      0,
      SHM_SIZE
   );

   if (pBuf == NULL) {
      fprintf(stderr, "Could not map view of file (%d).\n", GetLastError());

      CloseHandle(hMapFile);
      return 1;
   }

   // Write a message to the shared memory
   swprintf((wchar_t*)pBuf, L"Hello, this is a message from the first process.");

   // Fork a new process (not supported directly in Windows, you might use CreateProcess)
   // For simplicity, let's simulate a "fork" by waiting for user input
   printf("Press Enter to continue...\n");
```

```
    getchar();

    // Child process reads from shared memory and prints the message
    wprintf(L"Child process: %s\n", pBuf);

    // Unmap the file view
    UnmapViewOfFile(pBuf);

    // Close the file mapping object
    CloseHandle(hMapFile);

    return 0;
}
```