Memoria 10 Memoria condivisa WINDOWS

giovedì 20 novembre 2025 18:05

Vediamo ora come gestire la memoria condivisa in windows :

1. **Creazione**
   a.
   ```
   HANDLE CreateFileMapping(HANDLE hFile,
                   LPSECURITY_ATTRIBUTES
                                       lpAttributes,
                   DWORD flProtect,
                   DWORD dwMaximumSizeHigh,
                   DWORD dwMaximumSizeLow,
                   LPCTSTR lpName)
   ```

   Descrizione
   • invoca il mapping di un file in memoria

   Restituzione
   • handle al mapping del file

   b. Appena creato contiene tutti 0 o tutti NULL
   c. Ricordiamoci che la maniglia è locale al processo
2. **Apertura**
   a.
   ```
   HANDLE OpenFileMapping(DWORD dwDesiredAccess,
                          BOOL bInheritHandle,
                          LPCTSTR lpName)
   ```

   Descrizione
   • accede a un mapping di file esistente

   Restituzione
   • handle al mapping del file

   Parametri
   • dwDesiredAccess: modalità di accesso richiesta al mapping di file
   • bInheritHandle: specifica se l'handle restituito deve essere ereditato da processi figli
   • lpName: nome del mapping gia' esistente

   b.
   • hFile: handle ad un file aperto

   • lpAttributes: puntatore a una struttura SECURITY_ATTRIBUTES

   • flProtect: modalità di accesso al mapping del file (es. PAGE_READWRITE)

   c.
   • dwMaximumSizeHigh: dimensione massima del mapping del file (32 bit piu' significativi)

   • dwMaximumSizeLow: dimensione massima del mapping del file (32 bit meno significativi)

   • lpName: nome del mapping
3. **Mappatura**
   a.
   ```
   LPVOID MapViewOfFile(HANDLE hFileMappingObject,
                        DWORD dwDesiredAccess,
                        DWORD dwFileOffsetHigh,
                        DWORD dwFileOffsetLow,
                        SIZE_T dwNumberOfBytesToMap)
   ```

   Descrizione
   • innesta un mapping di file aperto nello spazio di indirizzamento del processo

   Restituzione
   • un puntatore all'area di memoria contenente il mapping del file, NULL in caso di fallimento

   b. Mappo il file mapping nell'address space
   c. Le due DWORD servo a specificare l' offset mappatura
4. **Un-mappatura**
   a.
   ```
   BOOL UnmapViewOfFile(LPCVOID lpBaseAddress)
   ```

   Descrizione
   • distacca un mapping di file aperto dallo spazio di indirizzamento del processo

   Restituzione
   • FALSE in caso di fallimento

   Parametri
   • lpBaseAddress: puntatore all'inizio di un'area di memoria innestata tramite MapViewofFile()

Vediamo un esempio :

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

#define DISP_ 20

#define Errore_(x) { puts(x); ExitProcess(1); }

char messaggio[256];

void scrittore(HANDLE mapping) {
        char *p;
        p = (char *) MapViewOfFile(mapping, FILE_MAP_WRITE,  0, 0, 0);
        if ( p == NULL ) Errore_("MapViewOfFile error");
        puts("Type the strings to post to file mapping ('quit' to close):");    fflush(stdout);
        do {
                scanf("%s", messaggio);
                strncpy(p, messaggio, DISP_);
                p += DISP_;
        } while( (strcmp(messaggio,"quit") != 0));

        ExitProcess(0);
}
```

Simile ad esempio di UNIX

```c
void lettore(HANDLE mapping) {
        char *p;
        p = (char *)MapViewOfFile(mapping, FILE_MAP_WRITE, 0, 0, 0);
        if (p == NULL) Errore_("MapViewOfFile error");
        printf("File mapping: \n");

        while ((strcmp(p, "quit") != 0)) {
                printf("%s\n", p);
                p += DISP_;
        }

        ExitProcess(0);
}
```

Simile ad esempio di UNIX

```c
int main(int argc, char *argv[]) {
      HANDLE mapping;
      BOOL newprocess;
      STARTUPINFO si;
      PROCESS_INFORMATION pi;

      if (argc == 1) {    /* Creazione/accesso segmento memoria condivisa */
            mapping = CreateFileMapping(INVALID_HANDLE_VALUE, NULL, PAGE_READWRITE, 0, 50000, "my_mapping");
            if (mapping == INVALID_HANDLE_VALUE) Errore_("Errore nella CreateFileMapping");

            /* genero il processo scrittore */
            memset(&si, 0, sizeof(si));
            memset(&pi, 0, sizeof(pi));
            si.cb = sizeof(si);
            newprocess =CreateProcess(".\\file-mapping.exe", ".\\file-mapping.exe scrittore", NULL, NULL, FALSE, NORMAL_PRIORITY_CLASS, NULL, NULL, &si,
&pi);
            if (newprocess == 0) {
                  Errore_("writer creation error");
            }
            WaitForSingleObject(pi.hProcess, INFINITE);
            memset(&si, 0, sizeof(si)); /* genero il processo lettore */
            memset(&pi, 0, sizeof(pi));
            si.cb = sizeof(si);
            newprocess = CreateProcess(".\\file-mapping.exe", ".\\file-mapping lettore", NULL, NULL, FALSE, NORMAL_PRIORITY_CLASS, NULL, NULL, &si,
&pi);
            if (newprocess == 0) {
                  Errore_("reader creation error");
            }
            WaitForSingleObject(pi.hProcess, INFINITE);
      }
      else
      if (argc == 2 && strcmp(argv[1], "scrittore") == 0){
            mapping = OpenFileMapping(FILE_MAP_WRITE, FALSE, "my_mapping"); /* Apro il file mapping */
            if (mapping == INVALID_HANDLE_VALUE) {
                  Errore_("OpenFileMapping error");
            }
            scrittore(mapping); /* chiamo la scrittura */
      }
      else
      if (argc == 2 && strcmp(argv[1], "lettore") == 0){
            mapping = OpenFileMapping(FILE_MAP_WRITE, FALSE, "my_mapping"); /* Apro il file mapping */
            if(mapping == INVALID_HANDLE_VALUE) {
                  Errore_("OpenFIleMapping error");
            }
            lettore(mapping); /* chiamo il lettore */
      }
      return(0);
}
```

La rimozione della shared memory in windows è automatica