```c
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#define SHMSZ 27

void server() {
        char c;
        int shmid;
        key_t key = 5678;
        char *shm, *s;

        // Creer le segment
        shmid = shmget(key, SHMSZ, IPC_CREAT | 0666);
        if(shmid < 0) { perror("shmget"); exit(1); }

        // Attacher le segment
        shm = shmat(shmid, NULL, 0);
        if (shm == (char *) -1) { perror("shmat"); exit(1); }

        // Mettre quelques choses dans la memoire pour l'autre processus
        s = shm;
        for (c = 'a'; c <= 'z'; c++) *s++ = c;
        *s = NULL;

        // On attend que le client lise en mettant en premier caractere '*'
        while (*shm != '*') sleep(1);

        shmdt((void*) shm);
        shmctl(shmid, IPC_RMID, NULL);
        exit(0);
}

void client() {
        int shmid;
        char *shm, *s;

        // Obtenir le segment "5678" cree par le serveur
        key_t key = 5678;
        shmid = shmget(key, SHMSZ, 0666);
        if(shmid < 0) { perror("shmget"); exit(1); }

        // Attacher le segment a notre espace de donnees
        shm = shmat(shmid, NULL, 0)
        if (shm == (char*) -1) { perror("shmat"); exit(1); }

        // Lire ce que le serveur a mis dans la memoire
        for (s = shm; *s != NULL; s++) putchar(*s);
        putchar('\n');

        // Changez le premier caractere du segment en '*' pour indiquer la lecture du segment
        *shm = '*';
        exit(0);
}
```

```c
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#define SHMSZ 27

int main(int, char**) {
        int shmid;
        char *shm, *s;

        // Obtenir le segment "5678" cree par le serveur
        key_t key = 5678;
        shmid = shmget(key, SHMSZ, 0666);
        if(shmid < 0) { perror("shmget"); exit(1); }

        // Attacher le segment a notre espace de donnees
        shm = shmat(shmid, NULL, 0)
        if (shm == (char*) -1) { perror("shmat"); exit(1); }

        // Lire ce que le serveur a mis dans la memoire
        for (s = shm; *s != NULL; s++) putchar(*s);
        putchar('\n');

        // Changez le premier caractere du segment en '*' pour indiquer la lecture du segment
        *shm = '*';
        exit(0);
}

```

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <sys/mman.h>

int main(int argc, char* argv[]) {
        int i= 0, fd = open(filename, O_RDWR, 0666);
        char* filename = "file.txt";
        struct stat st;

        stat(filename, &st);
        long fileSize = st.st_size;
        char* fileMap = (char*) mmap(NULL, fileSize, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);

        if(fileMap == (char*) MAP_FAILED) { perror("mmap"); exit(1); }
        close(fd);

        while(i < fileSize/2) {
                char c = fileMap[i];
                fileMap[i] = fileMap[fileSize - i-1];
                fileMap[fileSize - i-1] = c;
                i++;
        }

        printf("%s\n", fileMap);
        munmap((void*) fileMap, fileSize);
        return 0;
}

```

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <sys/mman.h>
#define FILESIZE 10

int main(int argc, char* argv[]) {
        int i= 0, fd = open("titi.dat", O_RDWR, 0666);

        int* fileMap = (int*) mmap(NULL, FILESIZE*sizeof(int), PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);

        if(fileMap == (int*) MAP_FAILED) { perror("mmap"); exit(1); }
        close(fd);

        while(1) {
                scanf("%d", &i);
                if(i == 99) break;
                if(i < 10 && i >= 0) fileMap[i] = fileMap[i]+1;
        }

        munmap((void*) fileMap, FILESIZE*sizeof(int));
        return 0;
}

```

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <sys/mman.h>
#define FILESIZE 10

int main(int argc, char* argv[]) {
        int i= 0, fd = open("titi.dat", O_RDONLY, 0666);

        int* fileMap = (int*) mmap(NULL, FILESIZE*sizeof(int), PROT_READ, MAP_PRIVATE, fd, 0);

        if(fileMap == (int*) MAP_FAILED) { perror("mmap"); exit(1); }
        close(fd);

        while(1) {
                scanf("%d", &i);
                if(i == 99) break;
                for(int j= 0; j< 10; j++) printf("\t%d\n", fileMap[j]);
        }

        munmap((void*) fileMap, FILESIZE*sizeof(int));
        return 0;
}
```

```c
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <sys/wait.h>
#include <unistd.h>

#define READSIZE 1024

int copyFile(int f1, int f2) {
        int totalByteRead= 0, lastByteRead= READSIZE;
        char buffer[READSIZE];

        while(lastByteRead == READSIZE) {
                lastByteRead = read(f1, buffer, READSIZE);
                write(f2, buffer, lastByteRead);
                totalByteRead += lastByteRead;
        }

        return totalByteRead;
}

int main(int argc, char* argv[]) {
        int file;
        id_t id = shmget(IPC_PRIVATE, sizeof(int), 0666);
        int* shmArea = (int*) shmat(id, NULL, 0);

        pid_t chld = fork();

        if(chld == 0)   file = open((argc > 2) ? argv[2] : "file2", O_RDONLY);
        else                    file = open((argc > 1) ? argv[1] : "file1", O_RDONLY);

        int byteCopied = copyFile(file, fileno(stdout));
        close(file);

        if(chld == 0) *shmArea = byteCopied;
        else
        {
                wait(NULL);
                printf("Bytes copied: %d & %d\n", byteCopied, *shmArea);

                shmdt((void*) shmArea);
                shmctl(id, IPC_RMID, NULL);
        }

        return 0;
}
```