```
//**********************************************************************
// File:    shmem.cc
// Author:  M. Thaler    18.01.2003
//
// Shared Memory
//**********************************************************************

#include "shmem.h"

//**********************************************************************
//* Static class variables

int    SharedMemory::shmSize = 0;
int    SharedMemory::shmID = 0;
void*  SharedMemory::shmAddr = NULL;
char*  SharedMemory::keyFilename = NULL;
int    SharedMemory::projectID = 0;

//**********************************************************************
//* Constructor & Destructor
//* if size != 0 then create and attach shared memory
//*    else get existing shared memory and attach it

SharedMemory::SharedMemory(int size) {
  shmSize = size;
  createSharedMemory();
}

SharedMemory::SharedMemory(int size,  const char* keyFile, int projID) {
  shmSize = size;
        keyFilename = (char *)keyFile;
        projectID = projID;
  createSharedMemory();
}

SharedMemory::~SharedMemory() {}  // do nothing

//**********************************************************************
//* return pointer to shared memory

void*
SharedMemory::getSharedMemory() {
  return SharedMemory::shmAddr;
}

//**********************************************************************
//* cleanup: remove shared memory and clean up file for key

void
SharedMemory::removeSharedMemory() {
  shmctl(shmID, IPC_RMID, NULL);  // remove shared memory
  if (keyFilename != NULL)
    unlink(keyFilename);  // delete key file
}

//**********************************************************************
//* create, get and attach the shared memory region

int
SharedMemory::createSharedMemory(void) {
  key_t key = IPC_PRIVATE;

  if (keyFilename == NULL) {
    cout << "shm: not yet implemented feature\n";
    exit(1);
  }

  // create key file, if not available
  int fd = open(keyFilename, O_RDWR | O_CREAT, 0770);
  close(fd);

  // get key by key file and ID
  key = ftok(keyFilename, projectID);

  // create or get shared memory
  if (shmSize > 0)
    shmID = shmget(key, shmSize, 0770 | IPC_CREAT);
  else
    shmID = shmget(key, shmSize, 0770);

  // attach shared memory to memory
  if (shmID >= 0) {
    shmAddr = shmat(shmID, NULL, 0);
    if (shmAddr == (void *)-1) {
      shmctl(shmID, IPC_RMID, NULL);
      shmID = -1;
    }
  }
  if (shmID < 0) {
    cout << "failed to allocate shared memory array\n";
    exit(-1);
  }
  return shmID;
}

//**********************************************************************

```