Programkod
Filsystem

Henrik Bäck
850611-6253
Mathias Andersson
850424-6292

Operating Systems DAVB01
VT 2005

Handledare: **Nils Dåverhög**
Hans Hedbom
Thijs Jan Holleboom

Inlämnad **2007-02-12**

lab5 arraifile.h

```
/*****************************************************************************/
/*                                                                           */
/*   Name: arrayfile.h                                                       */
/*                                                                           */
/*   Original Author: Hans Hedbom                                            */
/*                                                                           */
/*   Purpouse: Defines interface and structures for the SLF file system.     */
/*                                                                           */
/*   Revision History:                                                       */
/*   Version   Event            Author         Date                         */
/*   1.0b      Created          Hans Hedbom    030514 09:50 CET             */
/*   1.0c      Changed          Hans Hedbom    031009                       */
/*      Semantics for "open" is changed, to allow file creation              */
/*   1.0d      Changed          Per Strömgren  031009                       */
/*      Description of "name" in open delete                                 */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/

#ifndef _ARRAYFILE_
#define _ARRAYFILE_

/* Size definitions */

#define MAXSIZE 10
#define MAXFILESIZE 256
#define MAXOPENFILES 10
#define MAXNAMESIZE 10

/* Error definitions */

#define ERROR -1
#define OK 1

/* Node type definitions */

#define TYPE_DIR 1
#define TYPE_FILE 2



/* File system data structures */

/* Structure for storing file attributes */

typedef struct filenode{
  int type;
  int size; /* size of data */
  char name[MAXNAMESIZE];
  char data[MAXFILESIZE];
  }filenode;
```

```
/* Structure for storing directory attributes */

typedef struct dirnode{
  int type;
  int size; /* number of files in the directory */
  char name[MAXNAMESIZE];
  filenode files[MAXSIZE];
}dirnode;

/*Structure for storing information about open files */

typedef struct my_inode{
  int dir;
  int file;
}my_inode;

/* root mount point */

static dirnode root[MAXSIZE]={0};

/* File table of open files */

static my_inode filetable[MAXOPENFILES]={0};

typedef int my_fp;

/*************************************************************************/
/*                                                                       */
/*   Name:Create_Directory(char name[])                                  */
/*                                                                       */
/*   Purpouse: Creates a new directory in the file system                */
/*                                                                       */
/*   Parameters: name (The name given to the created directory)          */
/*                                                                       */
/*   Pre: True                                                           */
/*   Post:If root array is full or if the directory already exists Error */
/*        is returned and no directory is created otherwise a new directory */
/*        whit the name of parameter "name" is created.                  */
/*                                                                       */
/*************************************************************************/

int Create_Directory(char name[]);

/*************************************************************************/
/*                                                                       */
/*   Name:Open_File(char name[])  c                                      */
/*                                                                       */
/*   Purpouse: Opens the file with the name given in "name" and adds the */
/*             file to the file table.                                   */
/*                                                                       */
/*   Parameters: name. Form <directyory>/<file>                          */
/*              (name of the file to be opened)                          */
/*                                                                       */
/*   Pre: True                                                           */
/*   Post:If the file exists it is added to the file table and the index */
/*        at which the file is added is returned as the file pointer     */
/*        If the file does not exist, the file is created, and is added to  */
```

```
/*        the file table and the index at which the file is added is        */
/*        returned as the file pointer.                                     */
/*                                                                          */
/****************************************************************************/
my_fp Open_File(char name[]);

/****************************************************************************/
/*                                                                          */
/*   Name:Delete_File(char name[])                                          */
/*                                                                          */
/*   Purpouse: Deletes the file with the name "name"                        */
/*                                                                          */
/*   Parameters: name. Form <directyory>/<file>                             */
/*               (name of the file to be deleted)                           */
/*                                                                          */
/*   Pre: True                                                              */
/*   Post:If the file exists it is deleted from the directory               */
/*        otherwise noting is done.                                         */
/*                                                                          */
/****************************************************************************/
void Delete_File(char name[]);

/****************************************************************************/
/*                                                                          */
/*   Name:Write_File(my_fp file, char buffer[])                             */
/*                                                                          */
/*   Purpouse: Writes the content of buffer to the file pointed to by "file"*/
/*                                                                          */
/*   Parameters: file (The file table index of the file to write to)        */
/*               buffer( The text string to be written)                     */
/*                                                                          */
/*   Pre: File is open and the size of buffer is less than MAXFILESIZE      */
/*   Post:The content of buffer is written to the file. If the file is      */
/*        not empty any previous content is overwritten.                    */
/*                                                                          */
/****************************************************************************/
void Write_File(my_fp file, char buffer[]);

/****************************************************************************/
/*                                                                          */
/*   Name:Read_File(my_fp file, char buffer[])                              */
/*                                                                          */
/*   Purpouse: Writes the content of the file pointed to by "file" to buffer*/
/*                                                                          */
/*   Parameters: file (The file table index of the file to read from)       */
/*               buffer( The buffer to write the file content to )          */
/*                                                                          */
/*   Pre: File is open and the size of buffer is MAXFILESIZE                */
/*   Post:The content of the file is written to the buffer.                 */
/*                                                                          */
/****************************************************************************/
void Read_File(my_fp file, char buffer[]);


/****************************************************************************/
/*                                                                          */
/*   Name:Close_File(my_fp file)                                            */
```

```
/*                                                                          */
/*   Purpouse: Removes the entry at index "file" in the file table          */
/*                                                                          */
/*   Parameters: file (The index of the entry to be removed from file table)*/
/*                                                                          */
/*   Pre: File is open                                                      */
/*   Post:The files entry in the file table is removed.                     */
/*                                                                          */
/****************************************************************************/
void Close_File(my_fp file);

#endif
```

lab5_arraifile.cpp

```
#include <iostream>
#include <cstdlib>
#include <cstring>
#include "lab5_arrayfile.h"

//Filen vi ska testa med inkluderas för att slippa krångel
#include "lab5_test1.c"

using namespace std;

int Create_Directory(char name[])
{

  int raknare = 0;
  int wentOK = ERROR;
  bool exists = false;
  int pos = 0;

  pos = strcspn(name,"\0");

  if(pos <= MAXNAMESIZE)
    {
      if(name != "\0")
        {
          for(int j = 0;j<10;j++)
            {
              if(!strcmp(name,root[j].name))
                exists = true;
            }

          for(int i = 0;i<10;i++)
            {
              if(root[i].type != 0)
                raknare++;
            }

          if(raknare<10 && !exists)
            {
              root[raknare+1].type = TYPE_DIR;
              root[raknare+1].size = 0;
              strcpy(root[raknare].name,name);
```

```
              wentOK = OK;
            }
        }
      else
        {
          cout << "Ogiltigt katalognamn \n";
          wentOK = ERROR;
        }
    }
  else
    {
      cout << "För långt filnamn eller katalognamn\n";
      wentOK = ERROR;

      }

  return wentOK;

}

my_fp Open_File(char name[])
{

  char dir[MAXNAMESIZE+1];
  char file[MAXFILESIZE];
  int pos = 0;
  char* ptrF;
  my_fp returvardet;
  int katChk = -1;
  int filChk = -1;
  int bordFinns = -1;
  int nyFil = -1;

  pos = strcspn(name,"/");
  ptrF = strchr(name,'/');
  ptrF++;

  if(pos <= MAXNAMESIZE && strcspn(ptrF,"\0") < MAXFILESIZE)
    {

      strncpy(dir,name,pos);
      dir[pos]='\0';
      strcpy(file,ptrF);

      if(strcmp(dir,"\0") || strcmp(file,"\0"))
        {
          for(int i = 0;i<MAXSIZE;i++)
            {
              if(!strcmp(root[i].name,dir))
                katChk = i;
            }

          if(katChk != -1)
            {
              for(int i = 0;i<MAXSIZE;i++)
                {
                  if(!strcmp(root[katChk].files[i].name,file))
```

```
                filChk = i;
            }

          for(int i = 0;i<MAXOPENFILES;i++)
            {
              if(filetable[i].dir  == 0)
                bordFinns = i;
            }

          if(filChk != -1 && bordFinns != -1)
            {
              filetable[bordFinns].dir = katChk + 1;
              filetable[bordFinns].file = filChk + 1;
              returvardet = bordFinns;

            }
          else if(filChk == -1 && bordFinns != -1)
            {
               for(int i = 0;i<MAXSIZE;i++)
                {
                  if(root[katChk].files[i].type == 0)
                    nyFil = i;
                }

              if(nyFil != -1)
                {
                  root[katChk].files[nyFil].type = TYPE_FILE;
                  strcpy(root[katChk].files[nyFil].name,file);
                  root[katChk].files[nyFil].size = 0;

                  filetable[bordFinns].dir = katChk + 1;
                  filetable[bordFinns].file = nyFil + 1;
                  returvardet = bordFinns;
                }
              else
                {
                  cout << "\nKatalogen är full \n";
                  returvardet = ERROR;
                }
            }
          else
            {
              returvardet = ERROR;
            }
        }
      else
        {
          cout << "\nKatalogen finns inte. \n";
          returvardet = ERROR;
        }
    }
  else
    {
      cout << "\nFel indata \n";
      returvardet = ERROR;
    }
}
```

```
  else
    {
      cout << "\nFelaktigt katalog eller filnamn \n";
      returvardet = ERROR;
    }

  return returvardet;
}

void Delete_File(char name[])
{

  char dir[MAXNAMESIZE+1];
  char file[MAXFILESIZE];
  int pos = 0;
  char* ptrF;
  my_fp returvardet;
  int katChk = -1;
  int filChk = -1;
  bool oppen = false;

  pos = strcspn(name,"/");
  ptrF = strchr(name,'/');
  ptrF++;

  if(pos < MAXNAMESIZE && strcspn(ptrF,"\0") < MAXFILESIZE)
    {
      strncpy(dir,name,pos);
      dir[pos]='\0';
      strcpy(file,ptrF);

      if(dir != "\0" && file != "\0")
        {
          for(int i = 0;i<MAXSIZE;i++)
            {
              if(!strcmp(root[i].name,dir))
                katChk = i;
            }

          if(katChk != -1)
            {
              for(int i = 0;i<MAXSIZE;i++)
                {
                  if(!strcmp(root[katChk].files[i].name,file))
                     filChk = i;
                }

              for(int i = 0;i<MAXOPENFILES;i++)
                if(filetable[i].dir - 1 == katChk
                   && filetable[i].file - 1 == filChk)
                  oppen = true;

              if(filChk != -1 && oppen == false)
                {

                  root[katChk].files[filChk].type = 0;
                  root[katChk].files[filChk].size = 0;
```

```
                  strcpy(root[katChk].files[filChk].name,"0");

                }

              if(oppen)
                cout << "\nGår ej radera, filen är öppen\n";

            }
        }
    }
}




void Close_File(my_fp file)
{
  filetable[file].dir = 0;
  filetable[file].file = 0;
}

void Write_File(my_fp file, char buffer[])
{

  int fil,dir;

  dir = (filetable[file].dir) - 1;
  fil = (filetable[file].file) - 1;

  strcpy(root[dir].files[fil].data,buffer);

}

void Read_File(my_fp file, char buffer[])
{

  strcpy(buffer,root[(filetable[file].dir)-1].files[(filetable[file].file)
        -1].data);
}
```