renameMRI 1.0

Riku Klén
Turku PET Center

27.12.2004

Turku PET Centre Information Technology Report Series

Report TPCIT0010

Contents

1 What is renameMRI?

Program `renameMRI` is used to rename and sort DICOM 3.0 format MRI images. At the moment the program can be used only by administrator. In the future it will be automatical.

1.1 Usage

The program is used with command prompt in Windows operating system. It has to be called in the directory that contains only patient directories.

Calling `renameMRI incoming` (or `renamemri incoming`) will rename data of the directory `incoming` and sort it into specified directories.

Calling `renameMRI incoming r` (or `renamemri incoming r`) will only rename data of the directory `incoming`.

Calling `renameMRI incoming m` (or `renamemri incoming m`) will only sort data of the directory `incoming` into specified directories.

While files are renamed it is possible that a file with new name already exists. In this case a permission to overwrite the file is asked for each file. This can be avoided by calling the program with option `r` and adding a fourth parameter. The fourth parameter can be any word. For example calling `renameMRI incoming r o` (or `renamemri incoming r o`) will rename data of the directory `incoming` and not ask permission to overwriting.

1.2 Format of the Original Data

When the program is called the data has to be in standard DICOM format in the given directory. This means that the directory shall contain only patient directories. Each patient directory shall contain only DICOM directories, which contain only DICOM 3.0 format image files.

Example given. Let patient directories be patient1, patient2,... , DICOM directories be study1, study2,... and DICOM files be file1, file2,.... Now hierarchy of directories and files is shown in figure 1.

If data is not in this format the program cannot rename it. In some cases it cannot even sort it. If a DICOM directory contains more than one set of images none of them is renamed.

1.3 Format of the Sorted Data

The program renames all the directories, subdirectories and files contained in the given directory. Headers of DICOM files are read to find needed information

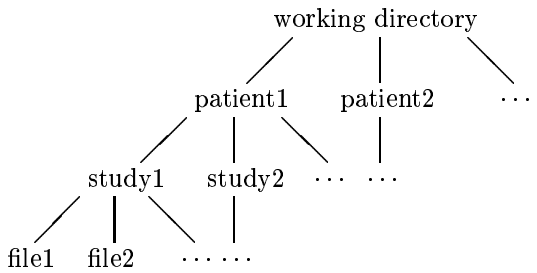

Figure 1: Hierarchy of directories and files of DICOM format. Working directory refers to the input directory.

for renaming. If no such data is available directories or files cannot be renamed.

Patient directories are named by patient name, patient id and study date (in form yyyymmdd). If some entry cannot be found, then it is left off the directory name. If none of them can be found, then patient directory cannot be renamed.

DICOM directories are named by patient name, patient id, study date (in form yyyymmdd), study time (in form hhmmss) and study description. If some entry cannot be found, then it is left off the directory name. If none of them can be found, then patient directory cannot be renamed.

DICOM files are renamed MR0001, MR0002, ..., MR9999, where number refers to the image number stored in the DICOM header. If image number cannot be found, then file is not renamed.

If a DICOM directory contains more than one set of images, then neither DICOM directory nor DICOM files are renamed.

Example given. If needed data is available, then renamed patient directories are in format

SURE_FIRST_MIDDLE_000000-0000_20040813

and renamed DICOM directories are in format

SURE_FIRST_MIDDLE_000000-0000_20040813_094312_DESCRIPTION,

where SURE refers to sure name, FIRST to first name and MIDDLE to the middle name.

Data is sorted into 4 directories, which are determined in function `move_mri()` in file `move_mri.c`. There is one directory for each MRI scanner (`c:/mri/GE_signa`, `c:/mri/Siemens` and `c:/mri/U_Picker`) and one directory for data that cannot be identified (`c:/mri/others`).

The program scans through the DICOM files of a patient folder. Once scanner data of a DICOM file header is identified, patient directory and all the

subdirectories and files are moved to the scanner specific directory.

2 Analysis of The Program `renameMRI` Version 1.0

The program is compiled with DJGPP in Windows 2000 operating system. It consists of 4 header files and 4 c-files. It uses libraries `lm` and `libpet` (functions determined in `swap.h`, version 2002-08-23).

2.1 File `move_data.c`

This file contains only one real function. It is used to move entire directories recursively. The other function is for test purposes.

2.1.1 Function `void test_data()`

If this function is called, then integer `MOVE_DATA` will get value 1 instead of initial value 0. Integer `MOVE_DATA` is used in function `move_data` to test, if some additional information should be printed.

2.1.2 Function `int move_data(char*, char*, int)`

The first input argument is a pointer to the name of file or directory to be moved. The second argument is a pointer to the directory where data is moved and the third argument is number of recursive calls done. It should be 0 for the initial call, 1 for the first recursive call, 2 for the next one and so on. Return value of the function is 0 if there was no problem and 1 otherwise.

If the first argument points to a file, then it is renamed using function `rename` from `stdio.h`. Otherwise new directory is created using function `mkdir` from `sys/stat.h`. All the files and directories inside the input directory are copied into the new directory by calling function `move_data` recursively. Once all data is copied the input directory is removed using function `remove` from `stdio.h`. To scan files and directories inside a directory functions of `dirent.h` are used.

2.2 File `move_mri.c`

This file contains only one real function. It is used to move DICOM 3.0 format MRI data. The other function is for test purposes.

2.2.1 Function `void test_move()`

If this function is called, then integer `TEST_MOVE` will get value 1 instead of initial value 0. Integer `TEST_MOVE` is used in function `move_mri` to test, if some additional information should be printed.

2.2.2 Function `void move_mri(char*)`

The only argument of this function is the name of the directory, which contains DICOM data. If it points to a file, then an error line is printed and nothing else is done.

The function uses functions of `dirent.h` and `read_dicom.h` to scan the files of given directory. First the files of the given directory are scanned. If no information about the scanner is found, then all subdirectories are scanned. If no information about the scanner is found, then the directory is moved into directory `c:/mri/others/` using function `move_data`. If scanner information was found, then the directory is moved into directory `c:/mri/GE_signa/`, `c:/mri/Siemens/` or `c:/mri/U_Picker/` using function `move_data`.

2.3 File `read_dicom.c`

This file contains 8 functions and a test function. Functions are used to read a DICOM 3.0 format MRI image header.

2.3.1 Function `void test_dicom()`

If this function is called, then integer `TEST_DICOM` will get value 1 instead of initial value 0. Integer `TEST_DICOM` is used in the other functions to test, if some additional information should be printed.

2.3.2 Function `static int convert(int)`

The only argument is an decimal integer representation of an ASCII character. The function returns an decimal integer representation of an ASCII character.

The function will convert all unknown characters (input is negative) into '#'. However character 'Ä' is converted to 'A', 'ä' to 'a', 'Ö' to 'O', 'ö' to 'o', 'Å' to 'A' and 'å' to 'a'. All known characters will remain the same.

2.3.3 Function `int image_number(char*)`

The only argument is a pointer to DICOM file name. The function returns the DICOM image number (which is always positive integer) or zero, if it couldn't

be found.

The function reads 5000 first characters of the input file, if possible. If tag (0020,0013) is found, then image number is read and returned. Otherwise 0 is returned.

2.3.4 Function `int patient_name(FILE*, char*)`

The first argument is a pointer to DICOM file name and the second one is a pointer to a string. The function copies the name of the patient from DICOM file into the input string, if possible. If the patient name was succesfully copied, then zero is returned. Otherwise one is returned.

The function reads 5000 first characters of the input file, if possible. If tag (0010,0010) is found, then patient name is read. Characters '^' and ' ' in the name are replaced with character '_' and function `convert` is called for each character of the name. The modified patient name is copied into the second argument. If the tag (0010,0010) was found, then 0 is returned. Otherwise 1 is returned.

2.3.5 Function `int patient_id(FILE*, char*)`

The first argument is a pointer to DICOM file name and the second one is a pointer to a string. The function copies the ID number of the patient from DICOM file into the input string, if possible. If the patient ID number was succesfully copied, then zero is returned. Otherwise one is returned.

The function reads 5000 first characters of the input file, if possible. If tag (0010,0020) is found, then patient ID number is read. Characters '^' and ' ' in the ID number are replaced with character '_'. The modified patient ID number is copied into the second argument. If the tag (0010,0020) was found, then 0 is returned. Otherwise 1 is returned.

2.3.6 Function `int study_date(FILE*, char*)`

The first argument is a pointer to DICOM file name and the second one is a pointer to a string. The function copies the study date from DICOM file into the input string, if possible. If the study date was succesfully copied, then zero is returned. Otherwise one is returned.

The function reads 5000 first characters of the input file, if possible. If tag (0008,0020) is found, then study date is copied into the second argument and 0 is returned. Otherwise 1 is returned.

2.3.7 Function `int image_time(FILE*, char*)`

The first argument is a pointer to DICOM file name and the second one is a pointer to a string. The function copies the image time from DICOM file into the input string, if possible. If the image time was succesfully copied, then zero is returned. Otherwise one is returned.

The function reads 5000 first characters of the input file, if possible. If tag (0008,0033) is found, then image time is copied into the second argument and 0 is returned. Otherwise 1 is returned.

2.3.8 Function `int study_description(FILE*, char*)`

The first argument is a pointer to DICOM file name and the second one is a pointer to a string. The function copies the study description from DICOM file into the input string, if possible. If the study description was succesfully copied, then zero is returned. Otherwise one is returned.

The function reads 5000 first characters of the input file, if possible. If tag (0008,1030) is found, then study description is read. Characters '^' and ' ' in the study description are replaced with character '_' and function `convert` is called for each character of the study description. The modified study description is copied into the second argument. If the tag (0008,1030) was found, then 0 is returned. Otherwise 1 is returned.

2.3.9 Function `int scanner_data(char*)`

The only argument is a pointer to DICOM file name. The function returns an integer depending on the name of the manufacturer. One is returned for GE Medical, two for Philips and three for Siemens. If manufacturer is none of these, then zero is returned.

The function reads 5000 first characters of the input file, if possible. If tag (0008,0070) is found, then 7 first letters of the manufacturer name is read. The read string is compared with known possibilities and a specified integer is returned.

2.4 File `rename.c`

This file contains a test function, a main function and two other functions. The main function uses the other two functions to rename and sort DICOM 3.0 data.

2.4.1 Function `void test()`

If this function is called, then integer `TEST` will get value 1 instead of initial value 0. It will also call test functions `test_dicom`, `test_data` and `test_move`. Integer `TEST` is used in the other functions to test, if some additional information should be printed.

2.4.2 Function `int rename_mri(char*, char*, int)`

The first argument is a pointer to the name of a orginal DICOM directory. The second argument is a pointer to the name of the new DICOM directory and the last one is permission to overwrite without asking. If the last argument is zero and the file exists, then it is overwritten without warning. Otherwise permission is asked for each file that already exist. The function renames files in the given DICOM directory and the DICOM directory in a specific way (see section 1.3). The function returns 0, if there was no problem, 1, if given DICOM directory couldn't be opened, 2, if the first DICOM image file couldn't be found, or 3, if there are more than one image sets in the DICOM directory.

The function uses functions of `dirent.h` and `read_dicom.h`. It first checks that there is only one image set in the given DICOM directory. Then it renames the DICOM files and asks permission to overwrite, if such file already exist and the third argument isn't 0. Finally the DICOM directory is renamed. Files and directories are renamed with function `rename` from `stdio.h`.

2.4.3 Function `int rename_folder(char*, int)`

The first argument is a pointer to the name of a orginal DICOM directory and the second argument is permission to overwrite without asking. If it is zero and the file exists, then it is overwritten without warning. Otherwise permission is asked for each file that already exist. The function renames data in the given patient directory and the patient directory in a specific way (see section 1.3). The function returns 0 if there were no problems and otherwise 1.

The function uses functions of `dirent.h` and function `rename_mri`. It renames first all subdirectories, which are suposed to be DICOM directories, using function `rename_mri`. Then it renames the patient directory using function `rename` from `stdio.h`.

2.4.4 Function `int main(int, char*)`

Main function is the actual program `renameMRI`, which is described in section 1.1. It uses functions of `dirent.h` and functions `move_mri` and `rename_folder`.

The function first chechks the input parameters. If there is something wrong with them, then a help text is printed. Otherwise data is renamed or sorted or both. A line for each succesful change is printed.