File logger

Below is an example of a file logger. You can build it as a dll and use in your projects.

filelogger.h

RFileLogger is a class which is not officially supported, but is widely used for logging as it is versatile and useful. It provides both instanced and static methods: instanced (i.e. non-static) methods are faster since you have to set up a client-server session to the Logger server only once, but static methods are also useful, since you don't need to maintain and share a singleton logger class. RFileLogger takes a wide variety of descriptors and other data fairly easily, including variable argument lists.

Logs are always created in C:\Logs\[chosen directory]\ on the device - as of S60 3rd Edition platform, this is the actual root of C: drive, not C:\Data which was the root used by the file manager, so you will need a tool like Y-Browser to access them and create the directories initially. Log files will not be created if the directories do not exist, which is useful to switch logging on or off.

Please note that the output produced by RFileLogger is not always reliable. For example, if the client program panics, then sometimes the log file doesn't contain each line to be logged up to the point the fatal error has occured.

```
#ifndef __FILELOGGER_H__
#define __FILELOGGER_H__

#include <f32file.h>

const TInt KMaxLogEntrySize = 400;

#define LOGCLOSE            CFileLogger::Close()
#define LOGARG(txt,parm...) {_LIT8(KTxt, txt); CFileLogger::Write(KTxt, parm);}
#define LOGTXT(txt)         {_LIT8(KTxt, txt); CFileLogger::Write((const TDesC8&)KTxt);}
#define LOGDES8(des)        CFileLogger::Write((const TDesC8&)des);
#define LOGDES16(des)       CFileLogger::Write(des);
#define LOGERR(txt,err)     if (err) LOGARG(txt, err)
#define LOGCALL(exp)        {LOGARG("Calling \"%s\"", #exp); exp; LOGARG("Call to \"%s\" passed.", #exp);}
#define LOGENTER            LOGARG("%s start", __PRETTY_FUNCTION__)
#define LOGEXIT             LOGARG("%s end", __PRETTY_FUNCTION__)
#define LOGMEM(ptr)         LOGARG("%s [0x%x]", #ptr, (TUint)ptr)

class CFileLogger : public CBase
{
public:
 IMPORT_C static void Write(const TDesC8& aText);
 IMPORT_C static void Write(const TDesC16& aText);
 IMPORT_C static void Write(TRefByValue<const TDesC8> aFmt,...);
 IMPORT_C static void Close();

private:
 CFileLogger();
 ~CFileLogger();
 TBool Construct();
 static CFileLogger* Logger();
 void DoWrite();
 TBool GetLogFileNameWithoutExt(TDes& aFileName);

private:
 RFs                     iFs;
 RFile                   iFile;
 TBuf8<KMaxLogEntrySize> iLogBuffer;
};

#endif // __FILELOGGER_H__
```

filelogger.cpp

```
#include "filelogger.h"

#include <bautils.H>
#include <pathinfo.h>

_LIT(KLogFolder, "logs\\");
_LIT(KLogFileExt, ".log");
_LIT(KOldFileExt, ".old");

const TUint8 KBackSlash = '\\';
_LIT8(KLineEnd, "\r\n");

const TInt KTimeRecordSize = 20;

_LIT8(KTimeFormat,"%04d-%02d-%02d %02d:%02d:%02d ");
_LIT8(KLogStart, "-=== New %S log ==--\r\n");

_LIT(KSquareBracket, "[");

CFileLogger::CFileLogger()
{
}

TBool CFileLogger::Construct()
{
 TInt err = iFs.Connect();
 if (!err)
 {
  TFileName logFileName;
  err = !GetLogFileNameWithoutExt(logFileName);
  if(!err)
  {
```


(C) Copyright Nokia 2013. All rights reserved.

```
    TFileName oldLogFileName(logFileName);
    oldLogFileName.Append(KOldFileExt);
    iFs.Delete(oldLogFileName);
    logFileName.Append(KLogFileExt);
    iFs.Rename(logFileName, oldLogFileName);
    err = iFile.Create(iFs, logFileName, EFileShareAny | EFileWrite);
    if(!err)
    {
     TInt pos(0);
     iFile.Seek(ESeekEnd, pos);
     TBuf8<KMaxFullName> appName;
     appName.Copy(BaflUtils::ExtractAppNameFromFullName(RThread().FullName()));
     iLogBuffer.AppendFormat(KLogStart, &appName);
     err = iFile.Write(iLogBuffer);
    }
   }
  }
  return (!err);
}

CFileLogger* CFileLogger::Logger()
{
 CFileLogger* logger = (CFileLogger*)Dll::Tls();
 if (!logger)
 {
  logger = new CFileLogger;
  if (logger)
  {
   if (logger->Construct())
   {
    Dll::SetTls(logger);
   }
   else
   {
    delete logger;
    logger = NULL;
   }
  }
 }
 return logger;
}

EXPORT_C void CFileLogger::Close()
{
 delete (CFileLogger*)Dll::Tls();
 Dll::FreeTls();
}

CFileLogger::~CFileLogger()
{
 iFile.Close();
 iFs.Close();
}

TBool CFileLogger::GetLogFileNameWithoutExt(TDes& aFileName)
{

#ifndef _DEBUG
// Phone target
// You must create this folder for the logfile to be written:
// E:\Logs
 aFileName.Copy(PathInfo::MemoryCardRootPath());
#else
// Emulator target
// You must create this folder for the logfile to be written:
// C:\Symbian\9.1\S60_3rd_MR_2\Epoc32\winscw\c\Logs
 TChar drive;
 iFs.DriveToChar(EDriveC, drive);
 aFileName.Append(drive);
 const TUint8 KColon = ':';
 aFileName.Append(KColon);
 aFileName.Append(KBackSlash);
#endif
 aFileName.Append(KLogFolder);
 TBool res = BaflUtils::FolderExists(iFs, aFileName);
 if (!res)
 {
  aFileName.Zero();

 }
        else
       {
  TPtrC fileName(BaflUtils::ExtractAppNameFromFullName(RThread().FullName()));
// The following code will search for a subfolder
// with the name of your process.
// If you want to use this subfolder,
// you must create it manually or the logfile will not be written.
 /*     TPtrC procName(RProcess().FullName());
  TPtrC folderName(TParsePtrC(procName.Left(procName.Find(KSquareBracket))).Name());
  aFileName.Append(folderName);
  aFileName.Append(KBackSlash);
        */
  aFileName.Append(fileName);
 }
 return res;
}

EXPORT_C void CFileLogger::Write(const TDesC8& aText)
{
 CFileLogger* logger = Logger();
 if(logger)
 {
  logger->iLogBuffer.Copy(aText);
  logger->DoWrite();
 }
}

EXPORT_C void CFileLogger::Write(const TDesC16& aText)
{
 CFileLogger* logger = Logger();
 if(logger)
 {
  logger->iLogBuffer.Copy(aText);
  logger->DoWrite();
 }
}

EXPORT_C void CFileLogger::Write(TRefByValue<const TDesC8> aFmt,...)
{
 CFileLogger* logger = Logger();
```


(C) Copyright Nokia 2013. All rights reserved.

```
 if (logger)
 {
  VA_LIST list;
  VA_START(list, aFmt);
  logger->iLogBuffer.FormatList(aFmt, list);
  logger->DoWrite();
  VA_END(list);
 }
}

void CFileLogger::DoWrite()
{
 if(iFile.SubSessionHandle())
 {
  TTime time;
  time.HomeTime();
  TDateTime dateTime;
  dateTime = time.DateTime();
  TBuf8<KTimeRecordSize> timeRecord;
  timeRecord.Format(KTimeFormat, dateTime.Year(), dateTime.Month()+1, dateTime.Day()+1, dateTime.Hour(), dateTime.Minute(), dateTime
  iLogBuffer.Insert(0, timeRecord);
  iLogBuffer.Append(KLineEnd);
  iFile.Write(iLogBuffer);
#ifdef _DEBUG
  iFile.Flush();
#endif // _DEBUG
 }
}
```

MMP file

You should link against the following libs:

```
LIBRARY       euser.lib
LIBRARY       efsrv.lib
LIBRARY       PlatformEnv.lib
LIBRARY       bafl.lib
```

Sample usage

```
// a simple log entry
LOGTXT("log entry");

// log entry with 3 params
LOGARG("params: %d, %u, %S", aTIntParam, aTUintParam, &aTDesC8Param);

// log an error. The log entry is written only if err != KErrNone
LOGERR("error : %d", err);

LOGDES8(aDes8); // log an 8-bit descriptor

LOGDES16(aDes16); // log an 16-bit descriptor

// write a log entry before the call, call the function
// and write an entry after it
LOGCALL(User::LeaveIfError(iFs.Connect()));

// Log memory address.
// The following line will produce an entry like "iMyPointer [0x1e1f4470]"
LOGMEM(iMyPointer);

// Write log entries when you enter a function
// and when you return from it.
// The following example writes to log the following:
// 2007-08-29 06:02:04 CMyClass::Foo(int) start
// 2007-08-29 06:02:04 CMyClass::Foo(int) end
TBool CMyClass::Foo(TInt aArg)
{
 LOGENTER;
 //...
 LOGEXIT;
 return ETrue;
}

// closes the log. Should be called before the program exits
LOGCLOSE;
```

Example FileLogger DLL Project File:FileLogger.zip

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.