Reading and writing files

RFile is the base API for file reading and writing, it is quite simple to use as following code sample shows:

Header required:

```
#include <f32file.h>
```

Library needed:

```
LIBRARY    efsrv.lib
```

Source:

```
HBufC8* ReadFromFileL(const TDesC& aFile)
{
 HBufC8* FilBuff(NULL);

 RFile ReadFil;

        /*Open file for reading */
 if(KErrNone == ReadFil.Open(CCoeEnv::Static()->FsSession(),aFile,EFileRead))
 {
  CleanupClosePushL(ReadFil);

  TInt FilSiz(0);
  if(ReadFil.Size(FilSiz) == KErrNone)
  {
   if(FilSiz > 0)
   {
    FilBuff = HBufC8::NewL(FilSiz);
    TPtr8 ReadPoint(FilBuff->Des());

    ReadFil.Read(ReadPoint,FilSiz);
   }
  }

  CleanupStack::PopAndDestroy(1);//ReadFile
 }

 return FilBuff;
}
```

With this function you can read the whole content of the file into a HBufC byte buffer. Note that the name has to be full name (drive+path+name+extension). One problem with this approach is that if the file is open it will not work, then instead you could use the RFs API directly as shown in this sample:

```
HBufC8* ReadOpenFileL(const TDesC& aFile)
{
 HBufC8* FilBuff(NULL);

 TEntry MyEntry;
 User::LeaveIfError(CCoeEnv::Static()->FsSession().Entry(aFile,MyEntry));
```

```
 if(MyEntry.iSize > 0)
 {
  FilBuff = HBufC8::NewL(MyEntry.iSize);
  TPtr8 ReadPoint(FilBuff->Des());

  CCoeEnv::Static()->FsSession().ReadFileSection(aFile,0,ReadPoint,FilSiz);
 }

 return FilBuff;
}
```

The `ReadFileSection` reads the file without opening the file. And to write a file, you could use the `RFile` API by doing something like this:

```
void WriteToFileL(const TDesC& aFile,const TDesC8& aData)
{
 RFile ReadFil;

 /* Open file if it exists, otherwise create it and write content in it */
 if(BaflUtils::FileExists(CCoeEnv::Static()->FsSession(),aFile))
  User::LeaveIfError(ReadFil.Open(CCoeEnv::Static()->FsSession(),aFile,EFileWrite));
 else
  User::LeaveIfError(ReadFil.Create(CCoeEnv::Static()->FsSession(),aFile,EFileWrite));

 CleanupClosePushL(ReadFil);

 TInt position = 0;
        /*Set cursor at the end of the file to write content at the end of file on each
successive call to write() method. */
 User::LeaveIfError(ReadFil.Seek(ESeekEnd, position));

        /*To write content in next line on successive call to write() method. */
        User::LeaveIfError(ReadFil.Write(_L8("\r\n")));

        /*Write actual content to file. */
 User::LeaveIfError(ReadFil.Write(aData));

 CleanupStack::PopAndDestroy(1);//ReadFile
}
```

This implementation appends to the existing file or creates new file if file does not exist. In case you want to replace already existing file, then you can use `RFs` API's Delete()-function to delete the file or Replace()-function from the `RFile`.