

List_Files_and_Directories

The following code snippet shows how to get list of files and directories from a given drive(example C:\,D:\ etc).

File rfsengine.h with declaration of CRFsEngine is a part of FileBrowser example. You can find it in 3rd FP1 SDK onward.

```
#include <e32base.h>
#include <e32cons.h>
#include <e32debug.h>
#include <f32file.h>
#include "rfsengine.h"

_LIT(KFileBrowseTest, "FileBrowser");
_LIT(KCDrive, "C:\\");
_LIT(KPermissionDenied, "Permission Denied");
_LIT(KBackSlash, "\\");

GLDEF_C TInt E32Main()
{
    // Create the cleanup stack
    CTrapCleanup cleanup = CTrapCleanup::New();
    (re$RAPD CallBrowseDirsL());
    // Destroy the cleanup stack
    delete cleanup;

    return 0;
}

LOCAL_C void CallBrowseDirsL()
{
    eofonsele::NewL(KFileBrowseTest, TSize(KConsFullScreen, KConsFullScreen));
    CleanupStack::PushK(console);

    ListDire(KCDrive);

    CleanupStack::PopAndDestroy(); // close console
}

// Recursive function to list files and directories
LOCAL_C void ListDirectoriesL(const TDesC& aPath)
{
    CRFsEngine engine = CRFsEngine::NewL();
    CleanupStack::PushK(engine);

    ::Debug(aPath);

    TInt result = engine->GetDirectoryAndFileList(aPath);

    if (result == KErrNone)
    {
        TInt dirCount = engine->DirectoryCount();
        TInt index = 0;

        for (index=0; index<dirCount; index++)
        {
            // Get directories
            HBufC * subDirPath = SubDirPathLC(aPath, engine->DirectoryName(index));
            ListDirectoriesL(*subDirPath);
            CleanupStack::PopAndDestroy(subDirPath);
        }
    }
}
```

List_Files_and_Directories

```
        // List files in the directory
        TInt fileCount      = engine->FileCount();
        index              = 0;
        for (; index < fileCount; index++)
        {
            RDebug::Print(engine->FileName(index));
        }
    }
else if (result == KErrPermissionDenied)
    {
        RDebug      ::Print(KPermissionDenied);
    }
else
    User::Leave(result);

CleanupStack::Destroy(engine);
}

// Allocates a heap buffer from aPath and aSubdir, with backslash terminator
LOCAL_C HBufC* SubDirPathLC(const TDesC& aPath, const TDesC& aSubDir)
{
    HBufC* subDirBuf = HBufC::NewLC(aPath.Length() + aSubDir.Length() + 1);
    TPtr subDirBuf = subDirBuf->Des();
    subDirBuf.Copy(aPath);
    subDirBuf.Append(aSubDir);
    subDirBuf.Append(KBackSlash);
return (subDirBuf);
}
```