



embedded-os.de  
a little world of RTOS and data-communication protocols

# *pC/TFS Reference*

## *V2.35a*

### Haftungsausschluß

Der Autor übernimmt keinerlei Haftung für durch diesen Code entstandene oder entstehende Schäden an Hard- und Software. Er versichert lediglich, daß er den Code vielfältigen Test´s auf unterschiedlicher Hardware unterzogen hat, um seinerseits keine Fehler bestehen zu wissen. Sollten dennoch Fehler auftauchen oder Vorschläge zur Verbesserung des Codes an den Autor weitergegeben werden, so ist dieser bestrebt, Fehler schnellstmöglich auszumerzen oder Vorschläge einzuarbeiten.

### liability exclusion

The author takes over no liability for through this code originated or emerging damages to hardware and software. He assures merely that he subjected the code of diverse tests on different hardware, about for his part no mistakes to know exists. Mistakes nevertheless should appear or suggestions are passed on at the author to the improvement of the code, so this is striving, mistakes fastest to wipe out or to incorporate suggestions.

Das Tiny-File-System basiert auf hirarchisch doppelt-verketteten Listen, wodurch keine Beschränkung in Zahl der Dateien innerhalb eines Verzeichnisses oder Dateigröße (max 4GB) vorliegt. Jede Datei kann dabei in fragmentierter oder unfragmentierter Form erstellt, verwaltet und gelöscht werden.

Das FileSystem verwaltet dabei alle Namen von Verzeichnissen/Dateien als STRING, das heißt auch Sonderzeichen wie Punkt, Leerzeichen und alle anderen gehören zum Namen des Elementes. Jedoch nicht zulässig sind zwei Punkte hintereinander bzw. ein oder mehrere Slashes im Namen.

Reservierte Names-Elemente:

- .. - ein Verzeichnis zurück
- / - am Anfang des Pfades: für ab ROOT
- / - innerhalb des Pfades: als Trennung der Verzeichnis/Datei-Namen

Das gesamte FileSystem arbeitet dabei Case-Sensitive !

Es sind aber keine Mechanismen des File-Sharings hinterlegt. Das heißt, eine Datei kann gleichzeitig von mehreren Usern zum Lesen aber nur von einem User zum Schreiben geöffnet werden.

Als File-Attribute stehen Read-Only, Write-Only und Hidden(System) zur Verfügung. Hidden-Einträge werden dabei bei einem TFS\_GetFirst.. oder TFS\_GetNext.. nicht angegeben, können aber direkt zugegriffen werden.

Das gesamte File-System arbeitet auf einen linearen Speicherbereich von maximal 4 GByte. Die Nutzung einer MMU ist dabei ohne weiteres möglich.

Eine Verwendung moderner Daten-Flashes mit Pagegrößen von 128..2048 Bytes kann mittels eines "ReadPageToBuffer-UpdateBuffer-ErasePage-WriteBufferToPage" Mechanismus jederzeit möglich gemacht werden, jedoch prüft das TinyFileSystem nicht auf optimiertes (zusammengefaßtes) PageWrite.

Des Weiteren beachtet das TinyFileSystem keine Hot-Spots (High-update files).

#### User-Functions:

TFS-Control:	Description
<a href="#">TFS_Init</a>	Initialisierung des File-Systems
<a href="#">TFS_GetRev</a>	Gibt Zeiger auf TFS-Revision zurück
<a href="#">TFS_Flush</a>	sichern des TFS als Image in einer LINUX-Datei
<a href="#">TFS_Format</a>	Formatieren des Laufwerkes

User:	Description
<a href="#">TFS_BecomeUser</a>	User anmelden
<a href="#">TFS_BecomeUserSubROOT</a>	User in einem Sub-DIR als User-ROOT anmelden
<a href="#">TFS_CloseUser</a>	User abmelden
<a href="#">TFS_GetFreeSize</a>	Gibt den Brutto-Freispeicher des Laufwerkes zurück

Directories:	Description
<a href="#">TFS_CreateDir</a>	Verzeichnis erstellen
<a href="#">TFS_RemoveDir</a>	Verzeichnis löschen
<a href="#">TFS_RenameDir</a>	Verzeichnis umbenennen
<a href="#">TFS_ChangeDir</a>	aktuellen Pfad ändern
<a href="#">TFS_ChangeDirTemp</a>	aktuellen Pfad temporär ändern

TFS_BackDirTemp	temporären Pfad zurücksetzen
TFS_GetCurrentDir	Gibt aktuellen Verzeichnis-Namen (ohne Pfad) zurück
TFS_GetCurrentPath	Gibt aktuellen Verzeichnis-Pfad ab ROOT zurück

Files:	Description
TFS_CreateFile	Datei erstellen
TFS_RemoveFile	Datei löschen
TFS_RenameFile	Datei umbenennen
TFS_MoveFile	Datei verschieben/bewegen
TFS_AttribFile	Attribute einer Datei ändern
TFS_GetFileAttrib	Gibt Attribute einer Datei zurück
TFS_ResizeFile	Dateigröße ändern
TFS_GetFileSize	Gibt Dateigröße zurück
TFS_OpenFile	Datei in "modi" öffnen
TFS_CloseFile	Datei schließen
TFS_SeekFile	Zeiger in geöffneter Datei absolut setzen
TFS_TellFile	Gibt den Zeiger in geöffnete Datei zurück
TFS_ExpandFile	offene Datei vergrößern
TFS_SetEOF	setzt EndOfFile in offener Datei an aktuellen R/W-Zeiger
TFS_ReadFile	Lesen aus Datei
TFS_WriteFile	Schreiben in Datei
TFS_WriteFileE	Schreiben in Datei, vergrößern dieser wenn nötig
TFS_GetErrNo	Fehlercode von Open, Read, Write ... lesen

Links:	Description
TFS_CreateLink	Link auf ein Verzeichnis oder eine Datei erstellen
TFS_RemoveLink	Link löschen
TFS_RenameLink	Link umbenennen
TFS_ReadLink	Gibt Namen des gelinkten Eintrages zurück

Entries:	Description
TFS_GetFirstName	Gibt Namen des ersten Eintrag im Verzeichnis zurück
TFS_GetNextName	Gibt Namen des nächsten Eintrages im Verzeichnis zurück

TFS_GetFirst	Gibt alle Infos des ersten Eintrag im Verzeichnis zurück
TFS_GetNext	Gibt alle Infos des nächsten Eintrages im Verzeichnis zurück

**Error-Codes:**

Name	Decimal_Value	Description
TFS_NO_ERR	0	no errors
TFS_USR_OVF	200	no user free
TFS_DBL_USER	201	double user
TFS_NO_USER	202	not a valid user
TFS_SUB_USER	203	this DIR is ROOT of a active user
TFS_NAME_EXIST	210	name of entry exist in this DIR
TFS_NOT_EXIST	211	DIR or FILE not exist
TFS_PATH_ERR	212	error on PATH
TFS_TMP_DIR	213	Temp-Dir is still used / not set
TFS_NO_FILE	214	error on PATH / no FILE-Name given
TFS_FILE_RO	215	file to open for writing is read-only
TFS_FILE_WO	216	file to open for reading is write-only
TFS_FILE_EOF	217	end-of-file while reading or writing
TFS_NOT_EMPTY	218	entry not empty
TFS_FILE_OPEN	219	current user have a opened file
TFS_NO_DATA	220	current file-length is zero
TFS_WRONG_PTR	221	offset into open file is wrong (or size for R/W)
TFS_LINKED	230	entry is linked
TFS_MAX_LINK	231	entry is max count linked
TFS_LINK_ERR	232	error in link-mechanism
TFS_NO_LINK	233	no link to an entry in link-entry
TFS_MEM_ERR	250	error in memory-manager
TFS_MEM_OVF	251	memory overflow
TFS_WR_PTR	252	user-buffer for write/read is in TFS-area
TFS_WRITE_ERR	253	memory write error
TFS_INVALID	254	invalid TFS-image (for LINUX-Host)

# allgemeines

---

## TFS\_Init

U08 TFS\_Init(void)

Initialisiert das File-System und installiert bei Verwendung eines RTOS die benötigte Mutex/Semaphore. Sollte das System als unformatiert erkannt werden, so wird TFS\_Format() ausgeführt.

Diese Funktion muß vor allen anderen TFS-Diensten bei der Systeminitialisierung einmal aufgerufen werden.

Bei Nutzung des Linux-HOSTs wird vor dem Formattest versucht ein bestehendes IMAGE aus einer Linux-Datei zu laden.

Der Speicher des FileSystems kann dabei auf zwei Wegen deklariert werden:

- für RAM-Laufwerk (compiler known memory):

```
#define TFS_SIZE      0x00010000
U08 TFS_START[TFS_SIZE];
```

- für EE / FLASH / RAM (compiler unknown memory):

```
#define TFS_START      0x00200000 // TFS startaddress
#define TFS_SIZE      0x00010000 // TFS size
```

## Parameters

none

## Return Value

TFS_NO_ERR	erfolgreich initialisiert
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung
TFS_MEM_OVF	kann ROOT nicht erstellen - Laufwerk voll
TFS_INVALID	IMAGE ungültig (nur bei Linux-HOST)

## Example

```
#define TFS_SIZE      0x00010000
U08 TFS_START[TFS_SIZE];

void main(void)
{
    U08 returnOk;

    OS_Init();
    .
    .
    returnOk=TFS_Init();
    .
}
```

```
.  
OS_Start();  
}
```

---

## TFS\_GetRev

```
void TFS_GetRev(U08 OS_HUGE **pointer)
```

Gibt einen Zeiger auf die TFS-Revision ( NULL-terminiertes ASCII-Array ) zurück.

### Parameters

<code>**pointer</code>	pointer to pointer will get the address of array
------------------------	--

### Return Value

none

### Example

```
void OS_FAR Task1(void *data)
{
    U08 OS_HUGE *Revision;

    .
    .
    while(1)
    {
        .
        TFS_GetRev(&Revision);
        .
    }
}
```

---

## TFS\_Flush

U08 TFS\_Flush(void)

*Nur bei Nutzung des Linux\_HOSTs  
Sichert das File-System als IMAGE in eine Linux-Datei.*

### Parameters

*none*

### Return Value

TFS_NO_ERR	Laufwerk gesichert
<i>from LINUX</i>	<i>see LINUX</i>

### Example

```
void main(void)
{
    U08  returnOk;

    .
    returnOk=TFS_Init();
    .
    .
    returnOk=TFS_Flush();
}
```

---

## TFS\_Format

U08 TFS\_Format(void)

Formatiert das TFS-Laufwerk und trägt den ROOT-Eintrag ein. Dabei darf kein User am File-System angemeldet sein.

### Parameters

none

### Return Value

TFS_NO_ERR	Laufwerk formatiert
TFS_USER	mindestens ein User ist angemeldet
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung
TFS_MEM_OVF	File-System zu klein für ROOT-Eintrag

### Example

```
void OS_FAR Task1(void *data)
{
    U08 returnOk;
    .
    .
    while(1)
    {
        .
        returnOk=TFS_Format();
        .
        .
    }
}
```

---

## TFS\_BecomeUser

U08 TFS\_BecomeUser(TFS\_USER OS\_HUGE \*TFSUser)

Legt einen neuen User an.

Diese Funktion initialisiert den User-Control-Block und trägt den neuen User in die interne Liste ein. Jeder Task kann sich nur einmal als User anmelden, anschließend stehen jedem User TFS\_HANDLES für Datei-Zugriffe zur Verfügung. Erst nach Anmeldung eines Tasks als User kann dieser Laufwerks- und Dateizugriffe aufrufen.

### Parameters

*TFSUser	pointer to user-control-block
----------	-------------------------------

### Return Value

TFS_NO_ERR	User erfolgreich angelegt
TFS_DBL_USER	dieser User ist bereits angemeldet
TFS_USR_OVF	es sind bereits TFS_maxUSER angemeldet

### Example

```
TFS_USER TFS_User1;

void OS_FAR Task1(void *data)
{
    U08 returnOk;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
    }
}
```

---

## TFS\_BecomeUserSubROOT

U08 TFS\_BecomeUserSubROOT(TFS\_USER OS\_HUGE \*TFSUser, U08 OS\_HUGE \*FullDirPath)

Legt einen neuen User an und sperrt ihn in einem Sub-Verzeichnis des Laufwerkes als dessen ROOT-Verzeichnis ein.

Diese Funktion initialisiert den User-Control-Block und trägt den neuen User in die interne Liste ein. Jeder Task kann sich nur einmal als User anmelden, anschließend stehen jedem User TFS\_HANDLES für Datei-Zugriffe zur Verfügung. Erst nach Anmeldung eines Tasks als User kann dieser Laufwerks- und Dateizugriffe aufrufen.

Ein solcher User kann das übergebene User-ROOT niemals in Richtung Laufwerks-ROOT mittels aller möglichen Pfadangaben wie ".." , "/" , "/Dir1" , oder ähnlich verlassen.

**WARNUNG!** Ist in diesem Sub-Tree ein Link auf einen anderen vom Laufwerks-ROOT ausgehenden Tree enthalten (angelegt durch einen Drive-ROOT User), so kann dieser User über diesen Link die Sub-Verzeichnis Einsperrung verlassen !

### Parameters

*TFSUser	pointer to user-control-block
*FullPathDir	vollständige Pfadangabe der Verzeichnisses ab Drive-ROOT

### Return Value

TFS_NO_ERR	User erfolgreich angelegt
TFS_DBL_USER	dieser User ist bereits angemeldet
TFS_USR_OVF	es sind bereits TFS_maxUSER angemeldet
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_NOT_EXIST	das Verzeichnis existiert nicht

### Example

```
TFS_USER TFS_User1;

void OS_FAR Task1(void *data)
{
    U08 returnOk;

    while(1)
    {
        returnOk=TFS_BecomeUserSubROOT(&TFS_User1, "/Dir2/Dir21/Dir214");

        // Sperrt den User in Dir214 als sein ROOT ein

    }
}
```



## TFS\_CloseUser

U08 TFS\_CloseUser(void)

Löscht einen eingetragenen User aus der internen Liste. Dieser User/Task muß erst erneut angemeldet werden, bevor Zugriffe auf das Laufwerk vom ihm akzeptiert werden.

### Parameters

none

### Return Value

TFS_NO_ERR	User erfolgreich abgemeldet
TFS_FILE_OPEN	User hat aktuell eine Datei geöffnet
TFS_NO_USER	User unbekannt

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_CloseUser();
        .
    }
}
```

---

## TFS\_GetFreeSize

TFS\_LONG TFS\_GetFreeSize(void)

Gibt den aktuellen Brutto-Freispeicher des TFS-Laufwerkes zurück.

### Parameters

none

### Return Value

Ist die Größe gleich NULL, so kann mittels TFS\_GetErrNo() nachfolgend der FehlerCode abgeholt werden.

TFS_NO_ERR	kein Frei-Speicher (Laufwerk voll)
TFS_NO_USER	User unbekannt
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung

### Example

```
TFS_USER TFS_User1;

void OS_FAR Task1(void *data)
{
    TFS_LONG  free_mem;
    U08       returnOk;
    .
    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        free_mem=TFS_GetFreeSize();
        if(!free_mem)
            returnOk=TFS_GetErrNo();
        .
        .
    }
}
```

---

# Directory and File-Handling

---

## TFS\_CreateDir

U08 TFS\_CreateDir(U08 OS\_HUGE \*Name)

Erstellt das angegebene Verzeichnis im aktuellen oder übergebenen Pfad. Die Pfadangabe kann dabei absolut oder relativ erfolgen. Extensions der Verzeichnisse gehören in diesem System mit zum Namen.

### Parameters

*Name	Directory-name [with path]
-------	----------------------------

### Return Value

TFS_NO_ERR	Verzeichnis erstellt
TFS_NO_USER	User unbekannt
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_NAME_EXIST	ein Verzeichnis mit gleichem Namen existiert bereits in diesem Verzeichnis
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung
TFS_MEM_OVF	Laufwerk voll

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_CreateDir("/usr");           // from ROOT
        .
        returnOk=TFS_CreateDir("../local/test.src"); // from one level back
        .
        returnOk=TFS_CreateDir("config.save");     // in actual directory
        .
        .
        returnOk=TFS_CloseUser();
        .
    }
}
```



## TFS\_RemoveDir

U08 TFS\_RemoveDir(U08 OS\_HUGE \*Name)

Löscht das angegebene Verzeichnis im aktuellen oder übergebenen Pfad. Die Pfadangabe kann dabei absolut oder relativ erfolgen. Das zu löschende Verzeichnis muß dabei leer sein. Desweiteren darf dieses Verzeichnis nicht gelinkt sein.

### Parameters

*Name	Directory-name [with path]
-------	----------------------------

### Return Value

TFS_NO_ERR	Verzeichnis gelöscht
TFS_NO_USER	User unbekannt
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_NOT_EMPTY	das Verzeichnis ist nicht leer
TFS_SUB_USER	das Verzeichnis ist ROOT eines aktiven Users
TFS_LINKED	das Verzeichnis ist durch einen anderen Eintrag gelinkt
TFS_NO_LINK	Verzeichnis ist ein Link der auf keinen Eintrag verweist oder dieser dies nicht registriert hat
TFS_LINK_ERR	Fehler in Link-Mechanismus
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        returnOk=TFS_RemoveDir("config.save");    // in actual directory
        .
    }
}
```



## TFS\_RenameDir

U08 TFS\_RenameDir(U08 OS\_HUGE \*Name, U08 OS\_HUGE \*NewName)

Ändert den Namen des angegebenen Verzeichnisses im aktuellen oder übergebenen Pfad. Die Pfadangabe kann dabei absolut oder relativ erfolgen. Eine Pfadangabe im neuen Namen wird nicht verarbeitet, d.h. das Verzeichnis kann dadurch nicht bewegt werden !

### Parameters

*Name	Directory-name [with path]
*NewName	new Directory-name (a path will ignored)

### Return Value

TFS_NO_ERR	Verzeichnis umbenannt
TFS_NO_USER	User unbekannt
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_NAME_EXIST	ein Verzeichnis mit gleichem Namen existiert bereits in diesem Verzeichnis
TFS_NOT_EXIST	Verzeichnis existiert nicht in diesem Verzeichnis
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_RenameDir("/usr", "user");    // in ROOT
        .
        .
    }
}
```

---

## TFS\_ChangeDir

U08 TFS\_ChangeDir(U08 OS\_HUGE \*Name)

wechselt das aktuelle Verzeichnis. Die Pfadangabe kann dabei absolut oder relativ erfolgen.

### Parameters

*Name	Directory-name [with path]
-------	----------------------------

### Return Value

TFS_NO_ERR	Verzeichnis gewechselt
TFS_NO_USER	User unbekannt
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_NOT_EXIST	das Verzeichnis existiert nicht

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_ChangeDir("../test.src");
        .
        .
        .
    }
}
```

---

## TFS\_ChangeDirTemp

U08 TFS\_ChangeDirTemp(U08 OS\_HUGE \*Name)

wechselt vorrübergehend (temporär) das aktuelle Verzeichnis. Die Pfadangabe kann dabei absolut oder relativ erfolgen. Das bisherige aktuelle Verzeichnis wird intern vermerkt. Diese Funktionalität kann nur ein Level per User verwendet werden.

### Parameters

*Name	Directory-name [with path]
-------	----------------------------

### Return Value

TFS_NO_ERR	Verzeichnis gewechselt
TFS_NO_USER	User unbekannt
TFS_TMP_DIR	Temp-Dir ist bereits in Verwendung
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_NOT_EXIST	das Verzeichnis existiert nicht

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;
    U08  Name[TFS_Name_SIZE];

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_ChangeDirTemp("../userfiles");
        if(returnOk == TFS_NO_ERR) {
            .
            returnOk=TFS_GetFirstName(Name);
            .
            do {
                .
                returnOk=TFS_GetNextName(Name);
                .
                .
            } while(returnOk == TFS_NO_ERR);
            .
            returnOk=TFS_BackDirTemp();
        }
        .
    }
}
```

}  
}



## TFS\_BackDirTemp

U08 TFS\_BackDirTemp(void)

wechselt zurück zum Verzeichnis vor dem vorübergehenden (temporären) Wechsel in ein anderes Verzeichnis. Das von TFS\_ChangeDirTemp vermerkte Originalverzeichnis wird wieder das aktuelle Verzeichnis.

### Parameters

none

### Return Value

TFS_NO_ERR	Verzeichnis gewechselt
TFS_TMP_DIR	Temp-Dir ist/wurde nicht gesetzt
TFS_NO_USER	User unbekannt

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;
    U08  Name[TFS_Name_SIZE];

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_ChangeDirTemp("../userfiles");
        if(returnOk == TFS_NO_ERR) {
            .
            returnOk=TFS_GetFirstName(Name);
            .
            .
            do {
                .
                returnOk=TFS_GetNextName(Name);
                .
                .
            } while(returnOk == TFS_NO_ERR);
            .
            returnOk=TFS_BackDirTemp();
        }
        .
        .
    }
}
```



## TFS\_GetCurrentDir

U08 TFS\_GetCurrentDir(U08 OS\_HUGE \*Name)

gibt das aktuelle Verzeichnis zurück. Es wird dabei nur der Verzeichnisname ohne Pfad zurückgegeben.

### Parameters

*Name	pointer to array for Directory-name
-------	-------------------------------------

### Return Value

TFS_NO_ERR	Verzeichnis gelesen
TFS_NO_USER	User unbekannt

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;
    U08  actDir[TFS_Name_SIZE];

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_GetCurrentDir(actDir);
        .
        .
        .
    }
}
```

---

## TFS\_GetCurrentPath

U08 TFS\_GetCurrentPath(U08 OS\_HUGE \*Path, TFS\_LONG maxlen)

gibt den kompletten Pfad des aktuellen Verzeichnisses zurück. Es wird dazu intern eine rekursive Funktion verwendet, um den Pfad ab (User-)ROOT bis einschließlich aktuellem Verzeichnisnamen zurückzugeben.

### Parameters

*Path	pointer to array for complete path
maxlen	max length of Path[]

### Return Value

TFS_NO_ERR	Pfad komplett gelesen
TFS_MEM_OVF	Pfad größer / gleich maxlen
TFS_NO_USER	User unbekannt

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;
    U08  actPath[1024];

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_GetCurrentPath(actPath, 1024);
        .
        .
    }
}
```

---

## TFS\_CreateFile

U08 TFS\_CreateFile(U08 OS\_HUGE \*Name, TFS\_ATTR Attr, TFS\_LONG size)

Erstellt die angegebene Datei im aktuellen oder übergebenen Pfad in angegebener Größe. Die Pfadangabe kann dabei absolut oder relativ erfolgen. Als Attribute können ReadOnly, WriteOnly und/oder Hidden angegeben werden. ACHTUNG! Dateien besitzen in diesem System keinen Typ.

### Parameters

*Name	File-name [with path]
Attr	Attributes of this file
size	size of file

### Return Value

TFS_NO_ERR	Datei erstellt
TFS_NO_USER	User unbekannt
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_NAME_EXIST	eine Datei mit gleichem Namen existiert bereits in diesem Verzeichnis
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung
TFS_MEM_OVF	Laufwerk voll

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_CreateFile("/file1", TFS_ATTR_RO, 100);
                                   // from ROOT
        .
        returnOk=TFS_CreateFile("../test.src/main.c", TFS_ATTR_SYS, 350);
                                   // from one level back
        .
        returnOk=TFS_CreateFile("makefile.mak", 0, 140);
    }
}
```

```
        : // in actual directory  
    }  
}
```

---

## TFS\_RemoveFile

U08 TFS\_RemoveFile(U08 OS\_HUGE \*Name)

Löscht die angegebene Datei im aktuellen oder übergebenen Pfad. Die Pfadangabe kann dabei absolut oder relativ erfolgen. Die zu löschende Datei darf dabei von keinem anderen Nutzer geöffnet sein. Desweiteren darf diese Datei nicht gelinkt sein.

### Parameters

*Name	File-name [with path]
-------	-----------------------

### Return Value

TFS_NO_ERR	Datei gelöscht
TFS_NO_USER	User unbekannt
TFS_FILE_OPEN	User (selber oder anderer) hat aktuell diese Datei geöffnet
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_LINKED	die Datei ist durch einen anderen Eintrag gelinkt
TFS_NO_LINK	Datei ist ein Link der auf keinen Eintrag verweist oder dieser dies nicht registriert hat
TFS_LINK_ERR	Fehler in Link-Mechanismus
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_RemoveFile("makefile.mak");    // in actual directory
        .
        .
    }
}
```

---

## TFS\_RenameFile

U08 TFS\_RenameFile(U08 OS\_HUGE \*OldName, U08 OS\_HUGE \*NewName)

Ändert den Namen der angegebenen Datei im aktuellen oder übergebenen Pfad. Die Pfadangabe kann dabei absolut oder relativ erfolgen. Die zu ändernde Datei darf dabei von keinem anderen Nutzer geöffnet sein.

Eine Pfadangabe im neuen Namen wird nicht verarbeitet, d.h. die Datei kann dadurch nicht bewegt werden !

### Parameters

*OldName	File-name [with path]
*NewName	new File-name (a path will ignored)

### Return Value

TFS_NO_ERR	Dateiname geändert
TFS_NO_USER	User unbekannt
TFS_NO_FILE	kein Datei-Name im Pfad oder als neuer Name angegeben
TFS_FILE_OPEN	User (selber oder anderer) hat aktuell diese Datei geöffnet
TFS_NAME_EXIST	eine Datei mit gleichem Namen existiert bereits in diesem Verzeichnis
TFS_NOT_EXIST	Datei existiert nicht in diesem Verzeichnis
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_MEM_OVF	Laufwerk voll

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_RenameFile("/test.src/main.c", "modul.c");
        .
        .
    }
}
```

}

---

## TFS\_MoveFile

U08 TFS\_MoveFile(U08 OS\_HUGE \*Name, U08 OS\_HUGE \*NewPath)

Bewegt die angegebenen Datei im aktuellen oder übergebenen Pfad in das angegebene Verzeichnis. Die Pfadangabe kann dabei absolut oder relativ erfolgen. Die zu bewegende Datei darf dabei von keinem Nutzer geöffnet sein.

Ein Dateiname im neuen Pfad wird nicht verarbeitet, d.h. die Datei kann dadurch nicht umbenannt werden !

### Parameters

*Name	File-name [with path]
*NewPath	new Path (a filename will ignored)

### Return Value

TFS_NO_ERR	Datei verschoben
TFS_NO_USER	User unbekannt
TFS_NO_FILE	kein Datei-Name im org.-Pfad angegeben
TFS_FILE_OPEN	User (selber oder anderer) hat aktuell diese Datei geöffnet
TFS_NAME_EXIST	eine Datei mit gleichem Namen existiert bereits im Zielverzeichnis
TFS_NOT_EXIST	Datei existiert nicht in diesem Verzeichnis
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_LINKED	die Datei ist durch einen anderen Eintrag gelinkt
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_MoveFile("/test.src/main.c", "../version.src");
        .
        .
    }
}
```

} }



## TFS\_AttribFile

U08 TFS\_AttribFile(U08 OS\_HUGE \*Name, TFS\_ATTR Attribs)

Ändert die Attribute der angegebenen Datei im aktuellen oder übergebenen Pfad. Die Pfadangabe kann dabei absolut oder relativ erfolgen. Die zu ändernde Datei darf dabei nicht geöffnet sein.

### Parameters

*Name	File-name [with path]
Attribs	new File-attributs

### Return Value

TFS_NO_ERR	Dateiattribute geändert
TFS_NO_USER	User unbekannt
TFS_NO_FILE	kein Datei-Name im Pfad angegeben
TFS_FILE_OPEN	aktuell ist diese Datei geöffnet
TFS_NOT_EXIST	Datei existiert nicht in diesem Verzeichnis
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_AttribFile("../TFS_err.log", TFS_ATTR_RO | TFS_ATTR_SYS);
        .
        .
    }
}
```

---

## TFS\_GetFileAttrib

TFS\_ATTR TFS\_GetFileAttrib(U08 OS\_HUGE \*Name)

Gibt die Attribute der angegebenen Datei im aktuellen oder übergebenen Pfad zurück. Die Pfadangabe kann dabei absolut oder relativ erfolgen.

### Parameters

*Name	File-name [with path]
-------	-----------------------

### Return Value

Sind die Attribute gleich -1, so kann mittels TFS\_GetErrNo() nachfolgend der Fehlercode abgeholt werden.

TFS_NO_ERR	Dateiattribute gelesen
TFS_NO_USER	User unbekannt
TFS_NO_FILE	kein Datei-Name im Pfad angegeben
TFS_NOT_EXIST	Datei existiert nicht in diesem Verzeichnis
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    TFS_ATTR attribs;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        attribs=TFS_GetFileAttrib("../TFS_err.log");
        if(attribs==(TFS_ATTR)(-1))
            returnOk=TFS_GetErrNo();
        .
        .
    }
}
```

---

## TFS\_ResizeFile

U08 TFS\_ResizeFile(U08 OS\_HUGE \*Name, TFS\_LONG newsize)

Ändert die Größe der angegebenen Datei im aktuellen oder übergebenen Pfad. Die Pfadangabe kann dabei absolut oder relativ erfolgen. Die zu ändernde Datei darf dabei nicht geöffnet sein.

### Parameters

*Name	File-name [with path]
newsize	new File-size

### Return Value

TFS_NO_ERR	Dateigröße geändert
TFS_NO_USER	User unbekannt
TFS_NO_FILE	kein Datei-Name im Pfad angegeben
TFS_FILE_OPEN	aktuell ist diese Datei geöffnet
TFS_NOT_EXIST	Datei existiert nicht in diesem Verzeichnis
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_MEM_OVF	Laufwerk voll
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        returnOk=TFS_ResizeFile("/Dir2/config.sys", 200);
        .
        .
    }
}
```

---

## TFS\_GetFileSize

```
TFS_LONG TFS_GetFileSize(U08 OS_HUGE *Name)
```

Gibt die Größe der angegebenen Datei im aktuellen oder übergebenen Pfad zurück. Die Pfadangabe kann dabei absolut oder relativ erfolgen.

### Parameters

*Name	File-name [with path]
-------	-----------------------

### Return Value

Ist die Größe gleich NULL, so kann mittels `TFS_GetErrNo()` nachfolgend der Fehlercode abgeholt werden.

TFS_NO_ERR	Dateigröße gelesen
TFS_NO_USER	User unbekannt
TFS_NO_FILE	kein Datei-Name im Pfad angegeben
TFS_NOT_EXIST	Datei existiert nicht in diesem Verzeichnis
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    TFS_LONG filesize;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        filesize=TFS_GetFileSize("/Dir2/config.sys");
        if(!filesize)
            returnOk=TFS_GetErrNo();
        .
        .
    }
}
```

---

# File-Access

---

## TFS\_OpenFile

TFS\_HANDLE TFS\_OpenFile(U08 OS\_HUGE \*Name, U08 Mode)

Öffnet die angegebenen Datei im aktuellen oder übergebenen Pfad. Die Pfadangabe kann dabei absolut oder relativ erfolgen. Für die nachfolgenden Zugriffe wird ein Handle zurückgegeben, unter diesem auf die Datei-Daten zugegriffen werden kann. Dieses Handle ist intern mit dem User referenziert und wird überprüft.

Sollte ein NULL-Handle zurück gegeben worden sein, so ist mittels TFS\_GetErrNo() der ERROR-Code abholbar.

### Access-Bedingungen:

Wenn eine Datei bereits (mehrfach) zum Lesen geöffnet ist, kann ein anderer User diese nicht zum Schreiben öffnen - wenn eine Datei zum Schreiben geöffnet ist, kann kein anderer User diese zum Lesen oder Schreiben öffnen.

## Parameters

*Name	File-name [with path]
mode	mode of access (ReadOnly/WriteOnly/ReadWrite)

## Return Value

Ist das Handle gleich NULL, so kann mittels TFS\_GetErrNo() nachfolgend der FehlerCode abgeholt werden.

TFS_NO_ERR	Datei geöffnet
TFS_NO_USER	User unbekannt
TFS_NO_FILE	kein Datei-Name im Pfad angegeben
TFS_FILE_OPEN	aktuell ist diese Datei geöffnet (siehe Access-Bedingungen)
TFS_FILE_RO	Datei ist ReadOnly und kann nicht "write" geöffnet werden
TFS_FILE_WO	Datei ist WriteOnly und kann nicht "read" geöffnet werden
TFS_NOT_EXIST	Datei existiert nicht in diesem Verzeichnis
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht

## Example

```
TFS_USER  TFS_User1;  
  
void OS_FAR Task1(void *data)  
{
```

```
U08      returnOk;
TFS_HANDLE  handl;

.
while(1)
{
    .
    returnOk=TFS_BecomeUser(&TFS_User1);
    .
    .
    handl=TFS_OpenFile("/Dir2/config.sys", TFS_ACC_RO);
    if(!handl)
        returnOk=TFS_GetErrNo();
    .
    .
}
}
```

---

## TFS\_CloseFile

U08 TFS\_CloseFile(TFS\_HANDLE Handl)

Schließt die aktuell geöffnete Datei.

### Parameters

Handl	File-Handle
-------	-------------

### Return Value

TFS_NO_ERR	Datei geschlossen
TFS_NO_FILE	Handle war nicht vergeben
TFS_NO_USER	User unbekannt

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    TFS_HANDLE  handl;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        handl=TFS_OpenFile("/Dir2/config.sys", TFS_ACC_RO);
        if(handl) {
            .
            returnOk=TFS_CloseFile(handl);
            .
        }
        .
    }
}
```

---

## TFS\_SeekFile

U08 TFS\_SeekFile(TFS\_LONG offset, TFS\_HANDLE Handl)

Setzt den R/W-Zeiger innerhalb der geöffneten Datei absolut.

### Parameters

offset	absolut pointer-position (in bytes)
Handl	File-Handle

### Return Value

TFS_NO_ERR	Zeiger in Datei gesetzt
TFS_NO_HAND	Handle ungültig
TFS_NO_USER	User unbekannt
TFS_NO_FILE	no file is open
TFS_NO_DATA	File has zero-lenght
TFS_WRONG_PTR	offset greater file-size

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    TFS_HANDLE  handl;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        handl=TFS_OpenFile("/Dir2/config.sys", TFS_ACC_RO);
        if(handl) {
            .
            returnOk=TFS_SeekFile(100, handl);
        }
        .
    }
}
```

---

## TFS\_TellFile

TFS\_LONG TFS\_TellFile(TFS\_HANDLE Handl)

Gibt den R/W-Zeiger der geöffneten Datei zurück.

### Parameters

Handl	File-Handle
-------	-------------

### Return Value

Ist die Position gleich NULL, so kann mittels TFS\_GetErrNo() nachfolgend der FehlerCode abgeholt werden.

TFS_NO_ERR	Zeiger in Datei gelesen
TFS_NO_HAND	Handle ungültig
TFS_NO_USER	User unbekannt
TFS_NO_FILE	no file is open
TFS_NO_DATA	File has zero-lenght

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    TFS_HANDLE  handl;
    TFS_LONG  posit;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        handl=TFS_OpenFile("/Dir2/config.sys", TFS_ACC_RO);
        if(handl) {
            .
            posit=TFS_TellFile(handl);
            if(!posit)
                returnOk=TFS_GetErrNo();
        }
        .
    }
}
```

## TFS\_ExpandFile

U08 TFS\_ExpandFile(TFS\_LONG addsize, TFS\_HANDLE Handl)

Vergrößert die geöffnete Datei.

### Parameters

addsize	File-size to add
Handl	File-Handle

### Return Value

TFS_NO_ERR	Dateigröße geändert
TFS_NO_HAND	Handle ungültig
TFS_NO_USER	User unbekannt
TFS_NO_FILE	no file is open
TFS_MEM_OVF	Laufwerk voll
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    TFS_HANDLE  handl;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        handl=TFS_OpenFile("/Dir2/config.sys", TFS_ACC_RO);
        if(handl) {
            .
            returnOk=TFS_ExpandFile(50, handl);
        }
        .
    }
}
```

---

## TFS\_SetEOF

U08 TFS\_SetEOF(TFS\_HANDLE Handl)

Setzt EndOfFile in offener Datei an aktuellen R/W-Zeiger.

### Parameters

Handl	File-Handl
-------	------------

### Return Value

TFS_NO_ERR	EndOfFile gesetzt
TFS_NO_HAND	Handle ungültig
TFS_NO_USER	User unbekannt
TFS_NO_FILE	no file is open
TFS_NO_DATA	File has zero-length
TFS_FILE_RO	File or Open-mode is Read-Only

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    TFS_HANDLE  handl;
    U08      writebuffer[]={ "TFS_Test_File R/W" };
    TFS_LONG  written;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        handl=TFS_OpenFile("/Dir2/config.sys", TFS_ACC_RW);
        if(handl) {
            written=TFS_WriteFile(writebuffer, strlen(writebuffer)-1, handl);
            if(written == strlen(writebuffer)-1)
                returnOk=TFS_SetEOF(handl);
            .
        }
        .
    }
}
```

---

## TFS\_ReadFile

```
TFS_LONG TFS_ReadFile(U08 OS_HUGE *dest, TFS_LONG size, TFS_HANDLE Handl)
```

Liest angegebene Zahl Bytes aus aktuell geöffneter Datei ab aktueller Position. Nach erfolgreichem Lesen steht der R/W-Zeiger hinter dem gelesenen Block.

Zurückgegeben wird die Zahl der gelesenen Bytes. Ist diese nicht identisch mit dem Aufruf, so kann mittels `TFS_GetErrNo()` der Fehlercode ermittelt werden.

### Parameters

*dest	Pointer to buffer where the bytes must written in
size	Bytes to read
Handl	File-Handle

### Return Value

Ist die zurückgegebene Länge ungleich der zu lesenden Bytes, so kann mittels `TFS_GetErrNo()` nachfolgend der FehlerCode abgeholt werden.

TFS_NO_ERR	Bytes aus Datei gelesen
TFS_NO_HAND	Handle ungültig
TFS_NO_USER	User unbekannt
TFS_NO_FILE	no file is open
TFS_NO_DATA	File has zero-lenght
TFS_FILE_EOF	End-Of-File
TFS_FILE_WO	File or Open-mode is Write-Only
TFS_WRONG_PTR	offset and/or size greater file-size

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    TFS_HANDLE  handl;
    U08      readbuffer[100];
    TFS_LONG  readed;

    .
    . . .
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
    }
}
```

```
handl=TFS_OpenFile("/Dir2/config.sys", TFS_ACC_RO);
if(handl) {
    returnOk=TFS_SeekFile(50, handl);
    readed=TFS_ReadFile(&readbuffer[0], 80, handl);
    if(readed != 80)
        returnOk=TFS_GetErrNo();
}
}
}
```

---

## TFS\_WriteFile

```
TFS_LONG TFS_WriteFile(U08 OS_HUGE *src, TFS_LONG size, TFS_HANDLE Handl)
```

Schreibt angegebene Zahl Bytes in aktuell geöffneten Datei ab aktueller Position. Nach erfolgreichem Schreiben steht der R/W-Zeiger hinter dem geschriebenen Block. Zurückgegeben wird die Zahl der geschriebenen Bytes. Ist diese nicht identisch mit dem Aufruf, so kann mittels `TFS_GetErrNo()` der Fehlercode ermittelt werden.

### Parameters

*src	Pointer to source-buffer
size	Bytes to write
Handl	File-Handle

### Return Value

Ist die zurückgegebene Länge ungleich der zu schreibenden Bytes, so kann mittels `TFS_GetErrNo()` nachfolgend der Fehlercode abgeholt werden.

TFS_NO_ERR	Bytes in Datei geschrieben
TFS_NO_HAND	Handle ungültig
TFS_NO_USER	User unbekannt
TFS_NO_FILE	no file is open
TFS_NO_DATA	File has zero-length
TFS_FILE_EOF	End-Of-File
TFS_FILE_RO	File or Open-mode is Read-Only
TFS_WRONG_PTR	actual offset plus size greater file-size

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    TFS_HANDLE  handl;
    U08      writebuffer[]={ "TFS_Test_File R/W" };
    TFS_LONG  written;

    .
    . . .
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
    }
}
```

```
.  
handl=TFS_OpenFile("/Dir2/config.sys", TFS_ACC_WO);  
if(handl) {  
    written=TFS_WriteFile(writebuffer, strlen(writebuffer)-1, handl);  
    if(written != strlen(writebuffer)-1)  
        returnOk=TFS_GetErrNo();  
    .  
    .  
    .  
    }  
    .  
}  
}
```

---

## TFS\_WriteFileE

```
TFS_LONG TFS_WriteFileE(U08 OS_HUGE *src, TFS_LONG size, TFS_HANDLE Handl)
```

Schreibt angegebene Zahl Bytes in aktuell geöffnete Datei ab aktueller Position. Nach erfolgreichem Schreiben steht der R/W-Zeiger hinter dem geschriebenen Block.  
Ist die Datei zu klein, so wird diese automatisch um die Differenz vergrößert. Dabei wird zuerst versucht das letzte Fragment der Datei zu vergrößern. Wenn dies nicht möglich ist, wird versucht, ein neues Fragment zu erstellen. Erst wenn auch dieses scheitert, wird aus den vorliegenden Freifragmenten der benötigte Mehrspeicher zusammengesetzt.  
Zurückgegeben wird die Zahl der geschriebenen Bytes. Ist diese nicht identisch mit dem Aufruf, so kann mittels `TFS_GetErrNo()` der Fehlercode ermittelt werden.

### Parameters

*src	Pointer to source-buffer
size	Bytes to write
Handl	File-Handle

### Return Value

Ist die zurückgegebene Länge ungleich der zu schreibenden Bytes, so kann mittels `TFS_GetErrNo()` nachfolgend der Fehlercode abgeholt werden.

TFS_NO_ERR	Bytes in Datei geschrieben
TFS_NO_HAND	Handle ungültig
TFS_NO_USER	User unbekannt
TFS_NO_FILE	no file is open
TFS_FILE_RO	File or Open-mode is Read-Only
TFS_MEM_OVF	Laufwerk voll
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    TFS_HANDLE  handl;
    U08      writebuffer[]={ "TFS_Test_File R/W(e)"};
    TFS_LONG  written;

    .
    .
    .
    {
        .
    }
}
```

```
returnOk=TFS_BecomeUser(&TFS_User1);
.
.
handl=TFS_OpenFile("/Dir2/config.sys", TFS_ACC_WO);
if(handl) {
    written=TFS_WriteFileE(writebuffer, strlen(writebuffer)-1, handl);
    if(written != strlen(writebuffer)-1)
        returnOk=TFS_GetErrNo();
    .
    .
    .
}
.
}
```

---

# Link-Handling

---

## TFS\_CreateLink

U08 TFS\_CreateLink(U08 OS\_HUGE \*OrgName, U08 OS\_HUGE \*Name)

Mit diesem Befehl kann ein Link auf eine Datei oder ein Verzeichnis erstellt werden. Dabei kann sich der Originaleintrag auch in einem anderen Zweig des Verzeichnisbaumes befinden. Es gelten für diesen Link die Attribute des gelinkten Eintrages.

### Parameters

*OrgName	File/Directory-name to link [with path]
*Name	Link-name [with path]

### Return Value

TFS_NO_ERR	Link erstellt
TFS_NO_USER	User unbekannt
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_NAME_EXIST	ein Eintrag mit gleichem Namen existiert bereits in diesem Verzeichnis
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung
TFS_MEM_OVF	Laufwerk voll
TFS_MAX_LINK	*OrgName ist TFS_MAX_Links gelinkt
TFS_LINK_ERR	Fehler im Linker

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    . . .
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        returnOk=TFS_CreateLink("/Dir2/config.sys", "../test.dir/linked.conf");
        .
        .
    }
}
```



## TFS\_RemoveLink

U08 TFS\_RemoveLink(U08 OS\_HUGE \*LinkName)

Mit diesem Befehl kann ein erstellter Link auf eine Datei oder ein Verzeichnis gelöscht werden.

--- ZU LÖSCHEN VON DIR / FILE: ---

Gelinkte Verzeichnisse / Dateien können solange nicht gelöscht werden, bis auch der letzte Link auf diesen Eintrag vorher gelöscht wurde !

Statt TFS\_RemoveLink() kann auch entsprechend dem Link-Typ (DIR/File) die zugehörige Stammfunktion TFS\_RemoveFile() bzw. TFS\_RemoveDir() durch den Nutzer aufgerufen werden.

### Parameters

*Name	Link-name [with path]
-------	-----------------------

### Return Value

TFS_NO_ERR	Link gelöscht
TFS_NO_USER	User unbekannt
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung
TFS_LINKED	*Name ist selber gelinkt
TFS_NO_LINK	Link zeigt auf kein Element bzw. Eintrag ist kein Link
TFS_LINK_ERR	Fehler im Linker

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    ...
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        returnOk=TFS_CreateLink("/Dir2/config.sys", "/test.dir/linked.conf");
        .
        .
        returnOk=TFS_RemoveLink("../test.dir/linked.conf");
    }
}
```

## TFS\_RenameLink

U08 TFS\_RenameLink(U08 OS\_HUGE \*OldName, U08 OS\_HUGE \*NewName)

Ändert den Namen des angegebenen Links im aktuellen oder übergebenen Pfad. Die Pfadangabe kann dabei absolut oder relativ erfolgen.

Eine Pfadangabe im neuen Namen wird nicht verarbeitet, d.h. der Link kann dadurch nicht bewegt werden !

### Parameters

*OldName	Link-name [with path]
*NewName	new Link-name (a path will ignored)

### Return Value

TFS_NO_ERR	Linkname geändert
TFS_NO_USER	User unbekannt
TFS_NAME_EXIST	ein Eintrag mit gleichem Namen existiert bereits in diesem Verzeichnis
TFS_NOT_EXIST	Link existiert nicht in diesem Verzeichnis
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_RenameLink("/test.dir/linked.conf", "config.link");
        .
        .
    }
}
```

---

## TFS\_ReadLink

U08 TFS\_ReadLink(U08 OS\_HUGE \*Name, U08 OS\_HUGE \*LinkName)

Gibt den Namen des gelinkten Eintrages zurück. Es wird dabei nur der Verzeichnis/File-Name ohne Pfad zurückgegeben.

### Parameters

*Name	Link-name [with path]
*LinkName	pointer to array for linked Entry-name

### Return Value

TFS_NO_ERR	Name gelesen
TFS_NO_USER	User unbekannt
TFS_PATH_ERR	ein Element der Pfadangabe existiert nicht
TFS_MEM_ERR	Fehler innerhalb der Speicherverwaltung
TFS_NO_LINK	Link zeigt auf kein Element bzw. Eintrag ist kein Link
TFS_LINK_ERR	Fehler im Linker

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;
    U08  Entry[TFS_Name_SIZE];

    .
    ...
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        returnOk=TFS_CreateLink("/Dir2/config.sys", "../test.dir/linked.conf");
        .
        returnOk=TFS_ReadLink("../test.dir/linked.conf", Entry);
        .
        returnOk=TFS_RemoveLink("../test.dir/linked.conf");
    }
}
```

---

# Entries lesen

---

## TFS\_GetFirstName

U08 TFS\_GetFirstName(U08 OS\_HUGE \*Name)

gibt den Namen des ersten Eintrages im aktuellen Verzeichnis zurück und setzt den "Next"-Counter des Users auf den Beginn des aktuellen Verzeichnisses. Es wird dabei nur der Verzeichnis- oder Dateiname ohne Pfad zurückgegeben.

### Parameters

*Name	pointer to array for Entry-name
-------	---------------------------------

### Return Value

TFS_NO_ERR	Eintrag gelesen
TFS_NOT_EXIST	kein Eintrag vorhanden
TFS_NO_USER	User unbekannt

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;
    U08  Name[TFS_Name_SIZE];

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_GetFirstName(Name);
        .
        .
    }
}
```

---

## TFS\_GetNextName

U08 TFS\_GetNextName(U08 OS\_HUGE \*Name)

gibt den Namen des nächsten Eintrages im aktuellen Verzeichnis zurück und setzt den "Next"-Counter des Users um einen Eintrag weiter. Es wird dabei nur der Verzeichnis- oder Dateiname ohne Pfad zurückgegeben.

### Parameters

*Name	pointer to array for Entry-name
-------	---------------------------------

### Return Value

TFS_NO_ERR	Eintrag gelesen
TFS_NOT_EXIST	kein weiterer Eintrag vorhanden
TFS_NO_USER	User unbekannt

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;
    U08  Name[TFS_Name_SIZE];

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        returnOk=TFS_GetFirstName(Name);
        .
        do {
            .
            returnOk=TFS_GetNextName(Name);
            .
        } while(returnOk == TFS_NO_ERR);
        .
    }
}
```

---

## TFS\_GetFirst

```
U08 TFS_GetFirst(TFS_GET OS_HUGE *get)
```

gibt alle relevanten Informationen des ersten Eintrages im aktuellen Verzeichnis in der übergebenen Struktur zurück und setzt den "Next"-Counter des Users auf den Beginn des aktuellen Verzeichnisses. folgende Informationen sind in der Struktur enthalten:

```
U08      Name[TFS_Name_SIZE]; // name of this entry
TFS_ATTR Attr;                // attributes of this entry
TFS_LONG Length;              // used length of this entry (0 if its a directory)
U08      Linked;               // not 0, if this entry is linked by an other entry
```

### Parameters

*get	pointer to GET struct
------	-----------------------

### Return Value

TFS_NO_ERR	Eintrag gelesen
TFS_NOT_EXIST	kein Eintrag vorhanden
TFS_NO_USER	User unbekannt

### Example

```
TFS_USER TFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    TFS_GET  get;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        .
        returnOk=TFS_GetFirst(&get);
        .
        .
        .
    }
}
```

---

## TFS\_GetNext

U08 TFS\_GetNext(TFS\_GET OS\_HUGE \*get)

gibt alle relevanten Informationen des nächsten Eintrages im aktuellen Verzeichnis in der übergebenen Struktur zurück und setzt den "Next"-Counter des Users um einen Eintrag weiter.

### Parameters

*get	pointer to GET struct
------	-----------------------

### Return Value

TFS_NO_ERR	Eintrag gelesen
TFS_NOT_EXIST	kein weiterer Eintrag vorhanden
TFS_NO_USER	User unbekannt

### Example

```
TFS_USER  TFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    TFS_GET  get;

    .
    while(1)
    {
        .
        returnOk=TFS_BecomeUser(&TFS_User1);
        .
        returnOk=TFS_GetFirst(&get);
        .
        do {
            .
            returnOk=TFS_GetNext(&get);
            .
        } while(returnOk == TFS_NO_ERR);
        .
    }
}
```

---

## Comments

---

## Comments

---