KAPITEL 5

von Michaël Van Canneyt

Einführung in die C-Bibliothek – Libc

- Ein Überblick über die Funktionalität der C-Bibliothek
- Dokumentation der C-Bibliothek
- Die C-Bibliotheksschnittstelle
- Das Dateisystem
- Erweiterte Dateioperationen
- Prozesse
- Signale
- Nicht blockierte I/O
- Weitere Funktionalitäten

Wenn Programme unter Linux eine Datei lesen oder eine Datei in einem Verzeichnis suchen, kommunizieren sie nicht direkt mit dem Linux-Kernel. Stattdessen verwenden sie die C-Bibliothek, die es erlaubt, diese Aufgaben einfacher zu lösen. Die C-Bibliothek enthält zahlreiche allgemeine C-Quelltexte zur Verwaltung von Strings, Zahlen und einer Menge weiterer Dinge. Vergleicht man damit die Gegebenheiten auf der Windowsplattform, verwaltet die C-Bibliothek die Funktionen, die dort in den dynamischen Bibliotheken »kernel32.dll« und »user32.dll« eingeführt wurden.

Anwendungen, die durch Kylix generiert wurden, unterscheiden sich in diesem Sinne nicht von C-Programmen. Die Low-level-Bausteine von Kylix-Anwendungen verwenden die Funktionen der C-Bibliothek.

Genauso verwenden Kylix-Programme die Qt-Bibliothek. Mit ihr werden verschiedene grafische Komponenten wie beispielsweise Fenster oder Comboboxen auf den Bildschirm gezeichnet. Einer Kylix-Anwendung ist nicht bekannt, wie sie eine Combobox zu zeichen hat. Dies wird durch die Qt-Bibliothek verwaltet. Qt verwendet ebenfalls die C-Bibliothek und zusätzlich noch die X-Bibliothek, die das eigentliche Zeichnen auf den Bildschirm ausführt. Die Qt-Bibliothek weiß zwar in Bezug auf Linien, Farben und so weiter, wie eine Combobox gezeichnet werden soll. Aber der eigentliche Zeichenvorgang ist ihr unbekannt. Er wird durch die X-Bibliothek verwaltet, die eigentlich die Combobox auf den Bildschirm zeichnet.

Grafisch sieht das so aus:

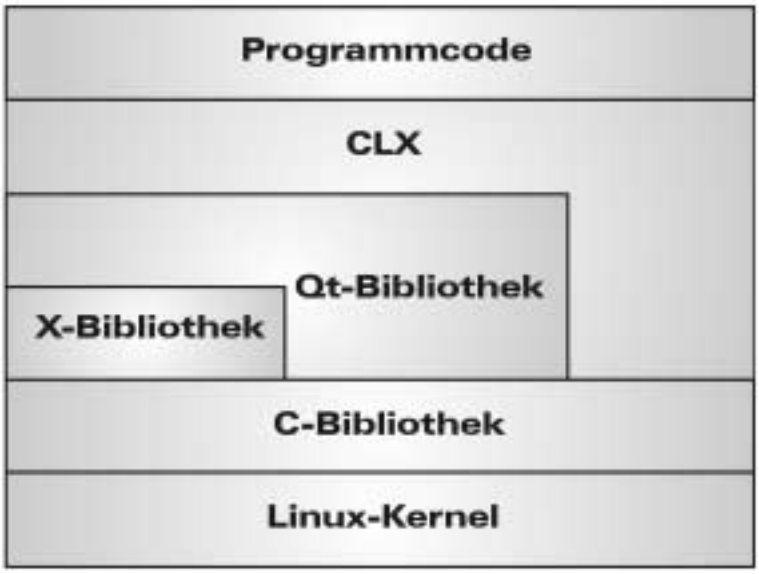

Abb. 5.1: Geschichtete Struktur

Abbildung 1 zeigt den funktionalen Aufbau von Kylix-Applikationen. An oberster Stelle residiert die CLX, die Klassenbibliothek von Kylix. Sie basiert wegen der GUI-Fähigkeit auf der Qt-Bibliothek, und auf der C-Bibliothek wegen aller anderen Funktionalitäten. Qt verwendet abwechselnd die X- und C-Bibliothek.

Diese hierarchische Struktur stellt sicher, daß Programme, die nur die CLX-Klassenbibliothek verwenden, durch Neukompilieren unter Windows beziehungsweise Linux ausgeführt werden können. Die Nachteile dieser Methode sind:

- Nicht alle Funktionalitäten der C-Bibliothek sind vorhanden; einige Konstrukte sind nicht portabel und infolge dessen in der Klassenbibliothek nicht vorhanden.
- Das Einführen zusätzlicher Schichten kann die Leistungsfähigkeit beeinflussen. Letzteres ist wahrscheinlich auf modernen Rechnersystemen weniger ausschlaggebend, aber beispielsweise könnten Daemon-Prozesse auf ausgelasteten Servern die Systemauslastung verbessern, indem sie versuchen, unnötigen Quelltext zu entfernen.

Wenn nun eine Anwendung Funktionen benötigt, die in der CLX-Bibliothek nicht vorhanden sind, ist es erforderlich, auf die C-Bibliothek zurückzugreifen.

5.1 Ein Überblick über die Funktionalität der C-Bibliothek

Die C-Bibliothek bietet eine große Auswahl an Funktionen, die von einem Programmierer für die Erstellung einer Anwendung unter Unix-Systemen benötigt werden. Einige Bereiche sind Teil der Standard-C-Bibliothek, andere sind Schnittstellen zum Linux- (oder allgemeiner: zum Unix-) Kernel. Erstere sind für einen Kylix-Programmierer nicht wirklich interessant, da die Funktionalität meist in den Units *System* und *SysUtils* vorhanden ist.

Der zweite Bereich ist von Bedeutung, wenn besondere Linux-Funktionen benötigt werden.

5.1.1 Routinen zur Dateibearbeitung

Die Libc verwaltet C-spezifische Routinen zum Umgang mit dem Dateisystem. Einige stammen aus der C-Standardbibliothek, andere sind einfach Schnittstellen zum Linux-Kernel. Die in der C-Bibliothek implementierte Dateisystemfunktionalität kann in folgende Kategorien unterteilt werden:

- **Dateien:** Die Funktionalität zum Öffnen von Dateien und zum Lesen von und Schreiben in Daten. Ein Teil dieser Funktionen ist wie in der C-Standardbibliothek implementiert. Der andere Teil ist einfach eine Schnittstelle zum Linux-Kernel. Auf diese Funktionen der Dateibehandlung der C-Bibliothek wird im folgenden nicht eingegangen, da sie sehr gut in den Units *System* und *SysUtils* von Kylix realisiert wurden.
- **Links:** Direkte und indirekte Verweise auf Dateien sind unter Linuxsystemen sehr gebräuchlich und werden viel umfassender als unter Windows verwendet. Die C-Bibliothek besitzt Kernel-Schnittstellen, um symbolische oder direkte Verweise in Dateisystemen zu erstellen und zu bearbeiten.
- **Zugriffserlaubnisse:** Die Verwaltung der Zugriffsrechte von Dateien im Linux-Dateisystem ist strenger als auf einem typischen Windowssystem. Systemdateien sind im allgemeinen schreibgeschützt (read-only) und für den normalen Benutzer nicht für Schreibzwecke zugelassen. Der Linux-Kernel und die C-Bibliothek stellen eine Funktion zur Verfügung, die die Zugriffsrechte auf Dateien überprüft und Ihrer Überwachung dient.
- **Weitere Attribute:** Dateien auf einem Linuxsystem haben einen anderen Aufbau des Attributsatzes als auf einem Windows-System; dazu zählen der Inode (eine einzigartige Nummer, die eine Datei in einem Dateisystem spezifiziert), verschiedene Zugriffszeiten und der Dateimodus.
- **Dateisperren:** Die Dateisperre hat unter Linux eine andere Semantik als unter Windows. An oberster Stelle stehen zwei Datei-Sperrsysteme, die sich gegenseitig nicht stören. Die C-Bibliothek verwaltet eine Kernelschnittstelle, die Dateien über beide Mechanismen sperren kann.
- **Verzeichnisse:** Die C-Bibliothek enthält Funktionen zum Anlegen und Entfernen von Verzeichnissen, außerdem Funktionen zum Lesen von Verzeichnisinhalten.
- **Geräte und Dateisysteme:** Ein Laufwerk und allgemein jedes Ein- und Ausgabegerät auf einer Linux-Maschine wird durch eine Gerätedatei im Verzeichnis »/dev« dargestellt. Laufwerke können überall im Verzeichnisbaum eingehängt werden. Die C-Bibliothek bietet eine Reihe von Funktionen zur Analyse von Dateisystemen und für das Einhängen von Dateisystemen in den Verzeichnisbaum.

5.1.2 Prozesse

Unix wurde von Grund auf als ein Mehrbenutzer- (Multi-User) und Multitasking-System entwickelt. Deshalb sind Informationen über laufende Prozesse und die Generierung anderer ein wichtiger Teil der C-Bibliothek und

des Linux-Kernels. Kylix selbst bietet keine Unterstützung für die Prozeßkontrolle. Dieser Teil der C-Bibliothek wurde nicht in Klassen verpackt.

In den folgenden Abschnitt werden folgende Themen behandelt:

- **Prozeß-IDs:** Jeder Prozeß besitzt eine einzigartige ID und einen einzigartigen Vorfahr. Diese Information kann ausschließlich gelesen und nicht gesetzt werden.
- **Erzeugen eines neuen Prozesses:** Um einen neuen Prozeß zu erzeugen, bedarf es zweier Schritte: Im ersten Schritt erstellt man eine Kopie des laufenden Prozesses, im zweiten Schritt ersetzt man die Kopie durch einen neuen Prozeß. Die C-Bibliothek bietet zahlreiche Funktionen zur Erzeugung eines neuen Prozesses.
- **Signale:** Der Kernel sendet bei bestimmten Ereignissen oder beim Auftreten von Fehlern Signale an ein Programm. Auch können Programme sich gegenseitig Signale zusenden. Die C-Bibliothek enthält Funktionen, um auf solche Signale zu antworten, den Empfang von Signalen zu deaktivieren und Funktionen zum Senden von Signalen.
- **Priorität:** Unter Linux können Prozessen Prioritätsstufen zugewiesen werden. Programme können ihre eigene Priorität erniedrigen oder sie wird durch den Systemadministrator erhöht. Die Funktionalität, um die Priorität eines Prozesses festzulegen, ist in der C-Bibliothek festgelegt.
- **Interprozeß-Kommunikation:** Prozesse können auf verschiedene Art und Weise miteinander kommunizieren. Für jede dieser Arten sieht die C-Bibliothek eine Kernel-Schnittstelle vor.
  - **Pipes:** Pipes sind Paare von Dateien im Sinne einer FIFO-Reihe (First-In/First-Out). Dieser Umstand findet in der Kommunikation zwischen Eltern- und Kind-Prozessen Anwendung.
  - **Semaphoren:** Semaphoren dienen dem Austausch von Zuständen zwischen Programmen. Sie werden beispielsweise dazu eingesetzt, mehrere Instanzen eines Programms zu erzeugen.
  - **Shared Memory: Über Shared Memory (**gemeinsamen Speicher) teilen sich zwei Programme einen Teil des Arbeitsspeichers, sie können auf diese Weise Daten gemeinsam nutzen.
  - **Nachrichten: Programme** senden sich gegenseitig Nachrichten und tauschen so Informationen über Ereignisse aus.

5.1.3 Systeminformationen

Durch den Kernel und die C-Bibliothek erhält man diverse Informationen über das laufende System.

- **Login-Name:** Der Login-Name des aktuell angemeldeten Benutzers kann abgerufen werden.
- **Shells:** Die Lage verschiedenartiger Shells.
- **Systemname und -informationen:** Der Name des Systems und die Kernelversion können ebenfalls ermittelt werden.

5.1.4 Weitere Funktionalitäten

Die C-Bibliothek umfaßt noch weitere Funktionen, die man im Großen und Ganzen unter folgenden Oberbegriffen zusamenfassen kann:

- **String-Funktionen:** Hierbei handelt es sich um die Funktionen der Stringbehandlung für die C-Bibliothek. Sie werden für Kylix-Programme nicht wirklich benötigt, da sie mit null-terminierten Strings arbeiten und die Kylix-Units *System* und *SysUtils* entsprechende Funktionen enthalten, die speziell für die Verarbeitung des ANSI-String-Formats geschrieben wurden.

  Diese Funktionen werden im folgenden nicht weiter beschrieben.
- **Verschlüsselung:** Unixsysteme sind standardgemäß mit einem Schema zur Ein-Wege-Verschlüsselung ausgerüstet, das in der C-Bibliothek definiert wird. Es wird beispielsweise in der Standard-Passwortbehandlung verwendet.
- **Kommandozeilenoptionen:** Linuxprogramme (insbesondere Kommandozeilenprogramme) verfügen im allgemeinen über viele Kommandozeilenoptionen. Die C-Bibliothek sieht Standardmechanismen für den einheitlichen Umgang mit den Kommandozeilenoptionen vor.
- **Dynamisch geladene Bibliotheken (DLL):** Die *Libc* enthält eine Schnittstelle zum Laden von Shared Libraries, die auch erlaubt, Adressen von Funktionen von dieser Bibliotheken zu beziehen.
- **Locale:** Der »Locale« bestimmt die von Linux verwendete Sprache. Die C-Bibliothek umfaßt Funktionen zur Feststellung, welche Locals durch den aktuellen Benutzer verwendet werden.
- **System Log:** Linux-Daemons oder Serverprozesse schreiben ihre Aktivitäten oft in das System-Log. Der System-Log-Daemon schreibt die Nachrichten unterschiedlicher Prozesse, abhängig von seiner Konfiguration, in verschiedene Dateien. Die C-Bibliothek besitzt ein einfach zu verwendendes Interface zum System-Log.
- **Sockets:** Sockets sind unter Unix-Systemen weit verbreitet. Sowohl TCP/IP-Sockets als auch Dateisystem-Sokkets können mit diesen Funktionen verarbeitet werden. Da Kylix Standard-Socket-Komponenten für

TCP/IP-Funktionen besitzt, gehe ich im folgenden nicht weiter darauf ein. Der Schwerpunkt meiner Ausführungen liegt stattdessen auf Dateisystem-Sockets, mit denen man normalerweise auf lokale Server zugreift. Beispielsweise verwendet der System-Log-Daemon ein Dateisystem-Socket.

5.2 Dokumentation der C-Bibliothek

Wurde die Dokumentation der C-Bibliothek auf dem System installiert, findet man sie in den Sektionen 2 und 3 der Man-Pages. Um eine Vorstellung davon zu erhalten, welche Funktionen verfügbar sind, ist es am einfachsten, das Programm »xman« zu öffnen – sofern installiert – und sich die Inhalte der Sektion 2 der Man-Pages anzeigen zu lassen. Im folgenden nun die Vorgehensweise:

1. Starten Sie *xman*. Falls Ihr Desktopsystem einen Menüeintrag für *xman* besitzt, können Sie auch diesen verwenden. Geben Sie in einem Shell-Fenster ein:

```
/usr/bin/X11/xman
```

Oder, falls sich das Verzeichnis mit dem X11-Binaries in der *PATH*-Umgebungsvariable befindet:

```
xman
```

Das startet ebenfalls den Manpage-Viewer für X.

2. Wenn *xman* gestartet ist, sollte es bei Ihnen wie in Abbildung 2 aussehen.

Abb. 5.2: Das Hauptfenster von xman

Drücken Sie auf den Button »Manual Page«, um ein Fenster für die »Man-Pages« zu öffnen.

3. Im Fenster »Manual pages« wählen Sie das Menüfeld »Sections« und »Section 2« oder »Section 3«. Das Anleitungsfenster zeigt jetzt eine Liste von verfügbaren Man-Pages an (vgl. Abbildung 3).

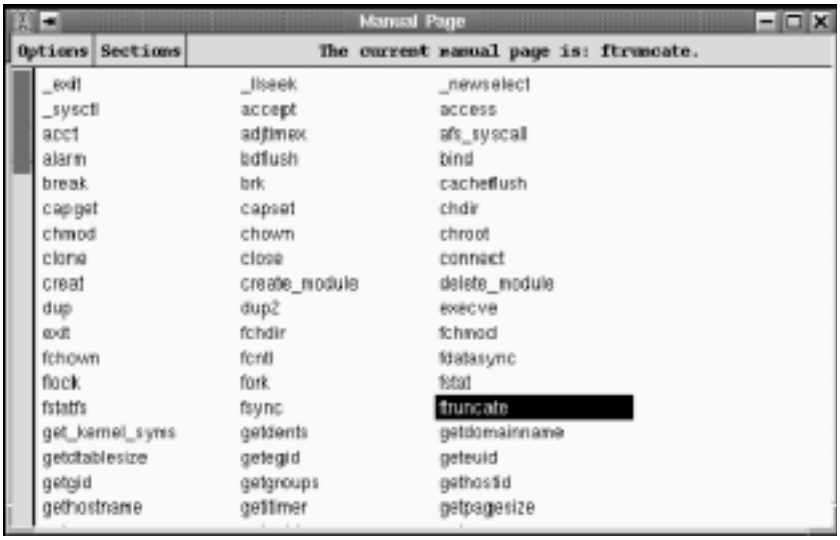

Abb. 5.3: Das Anleitungsfenster von xman zeigt den Inhalt eines Abschnitts an

Wird in der der Kylix-IDE auf einer Funktion, die in der Unit *Libc* (enthält alle Funktionen der C-Bibliothek) deklariert ist, [F1] gedrückt, startet Kylix ein X-Terminal mit den man-Pages für diese Funktion. Es ist nicht möglich, Informationen über andere Funktionen zu erhalten. Sie sollten *xman* also sinnvollerweise immer im Hintergrund laufen lassen.

5.3 Die C-Bibliotheksschnittstelle

Um die Funktionen in der C-Bibliothek nutzen zu können, muß Kylix wissen, welche Funktionen es überhaupt gibt. Die Funktionsdefinitionen der C-Bibliothek befinden sich in der Unit *Libc.pas* (case-sensitive!). Wenn man eine Funktion der C-Bibliothek verwenden möchte, muß diese Unit in der *uses*-Klausel des Programms oder der Unit, das/die Zugriff auf diese Funktion nehmen möchte, stehen.

Kylix fügt diese Unit normalerweise nicht automatisch in die *uses*-Klausel ein, man muß dies selbst tun.

Die meisten C-Funktionen verlangen null-terminierte Strings als Parameter. Verwendet man eine solche Funktion, erfordert dies gewöhnlich eine Typumwandlung von Ansi-String zu PChar. Um diese Typecasts überflüssig zu machen, werden in den nachfolgenden Kapiteln Funktionen definiert, die statt der null-terminierten Strings AnsiStrings als Parameter annehmen.

Viele C-Bibliotheksfunktionen liefern einen Integerwert zurück, um anzugeben, ob die Funktion erfolgreich durchgeführt wurde (Rückgabewert = 0) oder ob sich ein Fehler ereignet hat (Rückgabewert ungleich Null). Oft kommen diese Fehlerstatus-Funktionen direkt aus dem Linux-Kernel. Die Unit *UnixErrors* enthält zwei Routinen, die es erlauben, mit diesen Fehlern einheitlich umzugehen:

```
type
  EUnixOperationFailed = class(Exception)
  private
    FErrorCode: Integer;
  public
    constructor Create(AnErrorCode: LongInt);
    property ErrorCode: Integer Read FErrorCode;
  end;

function  StrError(Error: LongInt): String;
function  CheckUnixError(Error: Integer): Integer;
```

Die Funktion *StrError* wandelt eine Kernel-Fehlernummer (*parameterError*) in einen mehr oder weniger aussagekräftigen String um. Die Funktion *CheckUnixError* überprüft einen Fehlercode (Error). Falls dieser kleiner Null ist, wird eine Exception *EUnixOperationFailed* ausgelöst, andernfalls wird der Wert *errno* zurückgegeben.

Diese Funktion wird in den weiter unten entwickelten Programmen verwendet.

5.4 Das Dateisystem

Das Dateisystem von Linux unterscheidet sich grundsätzlich vom typischen Windows-Dateisystem – nicht nur in der Implementierung, sondern auch in der Anwendung. Jedes Programm, das eine große Anzahl an Dateiein- und ausgaben verwaltet, muß die Besonderheiten des Dateisystems berücksichtigen.

Für Einsteiger in Linux muß erwähnt werden, daß direkte und symbolische Verweise verstreut im gesamten Verzeichnisbaum eines Linuxsystems verwendet werden. Die C-Bibliothek bietet einige Funktionen, um

damit zurechtzukommen, meist sind dies Schnittstellen zu den Kernel-Funktionen.

Da Linux ein Mehrbenutzer-System ist, verfügt das Dateisystem über ein strenges System an Zugriffsberechtigungen. Auf einem durchschnittlichen Windows-System kann man aus den meisten Dateien lesen beziehungsweise auch hineinschreiben. In Linux ist ein Großteil des Systems für den Nicht-Administrator, also für den normalen Benutzer, der nicht als User root eingeloggt ist, schreibgeschützt. Normalerweise besitzt der Benutzer nur in seinem Home-Verzeichnis Lese- und Schreibrechte. Ein Programm sollte darauf Rücksicht nehmen und prüfen, ob es in ein Verzeichnis schreiben darf. Die C-Bibliothek besitzt Schnittstellen, um die Zugriffsrechte von Dateien zu ermitteln. Das Dateisystem unter Linux stellt mehr Attribute zu Verfügung als das Dateisystem von Windows. Die Standard-Units in Kylix emulieren das Verhalten von Windows, bieten aber nicht die ausgefeilten Informationen der C-Bibliothek. Im folgenden wird demonstriert, wie man diese zusätzlichen Daten ermittelt.

Die Suche nach Dateien wird in Kylix regulär mit dem Funktionspaar *FindFirst*/*FindNext* vorgenommen, in dem der Aufruf nativer Linux-Funktionen emuliert wird. Die nativen Linux-Funktionen zur Dateisuche werden weiter unten beschrieben, denn sie sind schneller und benötigen weniger Ressourcen.

Die Standardklasse *TFileStream* bietet eine rudimentäre Dateisperre, in der das Sperren von Dateien unter Windows emuliert wird. Eine fein granulierte Dateisperre ist aber nur mit den Funktionen der C-Bibliothek möglich, denn sie bietet mit der Funktion *fcntl* eine Kernelschnittstelle an.

Auf einer Linux-Maschine wird praktisch jede Hardware durch eine entsprechende Gerätedatei repräsentiert. Insbesondere Laufwerke werden durch eine bestimmte Gerätedatei dargestellt, die im Dateisystem an beliebiger Stelle eingebunden werden kann. Die C-Bibliothek bietet eine Schnittstelle zu den Kernel-Aufrufen, über die ein Laufwerk in das Dateisystem eingebunden wird. Ebenso wird eine Schnittstelle zum Erzeugen neuer Gerätedateien wie beispielsweise FIFOs bereitgestellt, .

Im folgenden wird jeder dieser Themenbereiche in einem kleinen, dem Dateiexplorer ähnlichen Programm demonstriert. Das Programm wird in verschiedene Zustände weiterentwickelt, indem in jedem Kapitel entsprechende weitere Funktionen hinzugefügt werden. Dies bedeutet, daß die Funktionalität dieses Programms sich im Laufe dieses Kapitels erweitern wird. Am Ende jedes Abschnitts werden wir ein komplett funktionsfähiges Programm vorliegen haben.

Die meisten Routinen, die sich in der C-Bibliothek befinden, werden wir in pascal-ähnliche Funktionen kapseln. Diese gekapselten Funktionen wer-

den dann in der Unit *UnixFileUtils* vereinigt. Wann immer eine Funktion der C-Bibliothek einen Wrapper in der Unit *UnixFileUtils* besitzt, wird dies im Text erwähnt.

Da der Entwickler oft mit der Ermittlung von Dateiattributen konfrontiert wird, ist dies unser erstes Thema.

5.4.1 Dateiattribute

Mit der Funktion *stat* erhält man die Attribute einer Datei. Sie ist wie folgt definiert:

```
function stat(FileName: PChar;
          var StatBuffer: TStatBuf): Integer; CDecl;
```

Die Funktion benötigt folgende Parameter:

FileName

Ein null-terminierter String, der den Dateinamen der Datei angibt, deren Attribute ermittelt werden sollen.

StatBuffer

Ein Record vom Typ *TStatBuf*, der mit den Attributen der Datei gefüllt wird, falls der Aufruf erfolgreich war. Der Record *TStatBuf* ist wie folgt definiert:

```
TStatBuf = record
  st_dev:     __dev_t;
  __pad1:     Word;
  st_ino:     __ino_t;
  st_mode:    __mode_t;
  st_nlink:   __nlink_t;
  st_uid:     __uid_t;
  st_gid:     __gid_t;
  st_rdev:    __dev_t;
  __pad2:     Word;
  st_size:    __off_t;
  st_blksize: __blksize_t;
  st_blocks:  __blkcnt_t;
  st_atime:   __time_t;
  __unused1:  LongWord;
  st_mtime:   __time_t;
  __unused2:  LongWord;
  st_ctime:   __time_t;
  __unused3:  LongWord;
  __unused4:  LongWord;
  __unused5:  LongWord;
end;
```