

Name:	Lehrgang: Betriebssysteme	Datum:
Arbeitsblatt Nr.	Das Linux-Dateisystem Second Extended Filesystem	Seite 1 von 2

Das **Second Extended Filesystem**, abgekürzt *ext2* oder auch *e2fs*, wurde für Linux entwickelt. Da Linux in der Anfangsphase unter Adrew Tanenbaums Betriebssystem Minix entwickelt wurde, fand dessen Dateisystem bei Linux Verwendung. Die Restriktionen von System V sind denen unter Minix ähnlich, daher wurde ein neues Dateisystem, eben *ext2*, entwickelt. Dieses Dateisystem ähnelt einem anderen unter UNIX weit verbreiteten Dateisystem, dem *BSD-Filesystem*.

Beim *ext2* wird eine Partition in mehrere Blockgruppen (in der Grafik sind lediglich zwei Blockgruppen dargestellt!) unterteilt. Im ersten physikalischen Block der Blockgruppe 1 befindet sich der **Bootblock** zum Laden des Betriebssystems. In den folgenden Blockgruppen bleibt der Bootblock jedoch leer.

Im zweiten Block befindet sich wiederum ein **Superblock**, der Meta-Informationen zum Dateisystem enthält. In den weiteren Blöcken befinden sich Kopien des Superblocks. Durch die Kopien existiert eine größere Ausfallsicherheit des Dateisystems.

Danach folgt ein **Blockgruppenskriptor** (D). Hierin sind Informationen über eine Blockgruppe gespeichert. Diese Informationen werden u.a. bei der Suche nach freien Blöcken und Inodes ausgewertet.

In der **Blockbelegungsbitmap** (BM) ist die Belegung der jeweiligen Blockgruppe gespeichert. Die **Inodebelegungsbitmap** (IM) enthält Informationen über die Auslastung der Inode Table.

Pro Block bzw. pro Inode ist hierfür ein Bit erforderlich. Dieses Bit wird als Schalter verwendet, der anzeigt, ob ein Block bzw. ein Inode-Eintrag verwendet wird oder nicht.

Den Belegungsbitmaps folgt die **Inode Table** für die jeweilige Blockgruppe.

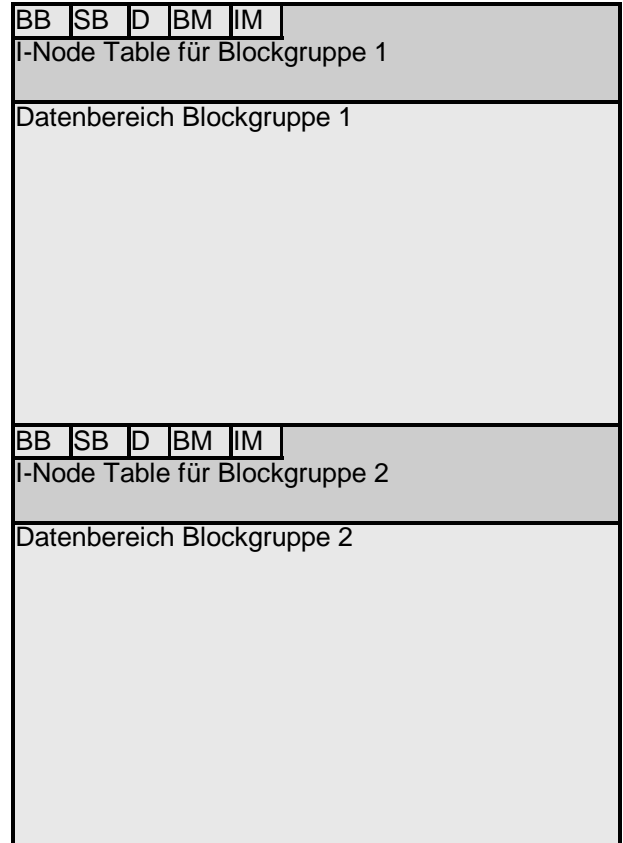
Die Größe der Blockgruppen wird durch die Tatsache beschränkt, daß die Inode- und die Blockbitmaps der Blockgruppe in jeweils einen Datenblock passen müssen. Bei einer Blockgröße von 1KByte können in einer Blockgruppe also maximal 8K Inodes bzw. 8K Blöcke als verwendet oder verfügbar gekennzeichnet werden (1 Bit pro Block).

Wird die Blockgröße geändert –möglich sind bei *ext2*-Filesystem Blockgrößen von 1K, 2K oder 4K- liegt das Limit bei entsprechend 16K oder 32K Objekten. Bei einer Blockgröße von 1K werden also Blockgruppen in der Größe von 8MByte erzeugt –8192 Blöcke zu jeweils 1024 Byte.

Neue Dateien versucht *ext2* in dem Datenblock zu erzeugen, in dem auch das Verzeichnis beheimatet ist, in dem die Dateien abgelegt werden sollen. Die Datenblöcke einer Datei oder eines Verzeichnisses versucht *ext2* in derselben Blockgruppe zu belegen, in der die Inode der Datei bzw. des Verzeichnisses liegt. Unter Berücksichtigung der Belegung in den einzelnen Blockgruppen wählt *ext2* unter den Gruppen mit vielen freien Inodes diejenige aus, die die meisten freien Datenblöcke aufweist. Durch diese Maßnahmen wird eine möglichst gleichmäßige Auslastung der einzelnen Blockgruppen und eine schnelle Erreichbarkeit der Datenblöcke erreicht.

Das Filesystem *ext2* unternimmt weitere Maßnahmen, um die Speicherung von und den Zugriff auf Daten zu optimieren.

- Beim Anlegen neuer Dateien wird der erste freie Datenblock als Ziel vorgegeben
- Beim Vergrößern von Dateien, wird der zuletzt von dieser Datei belegte Block angegeben
- Zusätzlich werden bei Anforderung eines weiteren Blocks bis zu acht zusammenhängende Blöcke „vorbestellt“. Nachfolgende Anforderungen von Blöcken für diese Datei werden dann aus diesem Pool vorbestellter Blöcke befriedigt. Die Vorbestellung wird u.a. aufgegeben, wenn die Datei geschlossen wird
- Beim Lesen von Dateien versucht *ext2* vorausschauend Leseanforderungen durchzuführen, um den Buffer Cache zu laden (*read ahead*). Außerdem versucht Linux aufeinanderfolgende Leseanforderun-



Name:	Lehrgang: Betriebssysteme	Datum:
Arbeitsblatt Nr.	Das Linux-Dateisystem Second Extended Filesystem	Seite 2 von 2

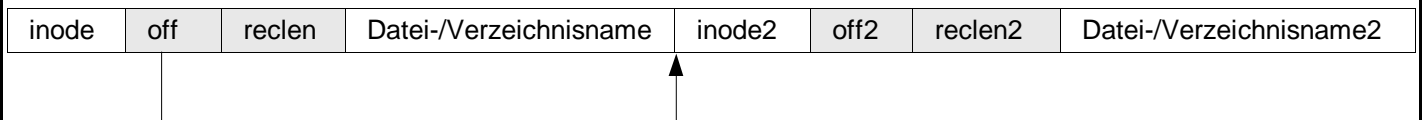
Die Inode-Struktur von ext2 ist 128 Byte groß.

```

/*
 * Constants relative to the data blocks
 */
#define EXT2_NDIR_BLOCKS 12
#define EXT2_IND_BLOCK EXT2_NDIR_BLOCKS
#define EXT2_DIND_BLOCK (EXT2_IND_BLOCK + 1)
#define EXT2_TIND_BLOCK (EXT2_DIND_BLOCK + 1)
#define EXT2_N_BLOCKS (EXT2_TIND_BLOCK + 1)
/*
 * Structure of an inode on the disk
 */
struct ext2_inode {
    __u16 i_mode; /* File mode */
    __u16 i_uid; /* Owner Uid */
    __u32 i_size; /* Size in bytes */
    __u32 i_atime; /* Access time */
    __u32 i_ctime; /* Creation time */
    __u32 i_mtime; /* Modification time */
    __u32 i_dtime; /* Deletion Time */
    __u16 i_gid; /* Group Id */
    __u16 i_links_count; /* Links count */
    __u32 i_blocks; /* Blocks count */
    __u32 i_flags; /* File flags (chflags) */
    /* gekürzt */
    __u32 i_block[EXT2_N_BLOCKS]; /* Pointers to blocks */
    __u32 i_version; /* File version (for NFS) */
    __u32 i_file_acl; /* File ACL, unused */
    __u32 i_dir_acl; /* Directory ACL, unused */
    __u32 i_faddr; /* Fragment address, unused */
    /* gekürzt */
};
    
```

**Datei- und Verzeichnisnamen**

Das ext2-Filesystem läßt bis zu 255 Zeichen lange Datei- bzw. Verzeichnisnamen zu. Die Verwaltung ist gegenüber dem System V Filesystem modifiziert, da ansonsten die Speicherung der Dateinamen sehr uneffizient wäre. Die Organisation erfolgt daher in der nachfolgend beschriebenen Weise:



Es wird eine Struktur verwendet, in der die Namenseinträge eine variable Länge haben können. Das Feld **inode** enthält die Inode-Nummer zum Verzeichniseintrag. Das Feld **off** gibt an, wie lang der vollständige Eintrag ist, und das Feld **reclen** gibt die Länge des gespeicherten Namens an.

**Weitere Änderungen gegenüber System V**

Bei ext2 sind sogenannte „symbolische Verweise“ möglich, die auch die Grenzen eines Dateisystems überspringen können.

Außerdem existiert eine Quota-Verwaltung. Es ist möglich, *harte* und *weiche* Limits zu vergeben, die durch das System ext2 überwacht werden.