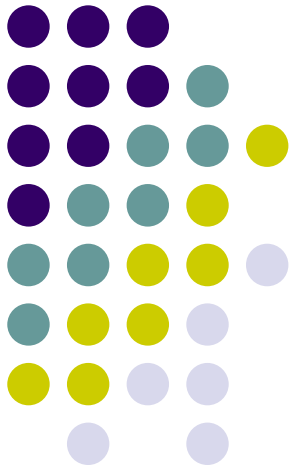
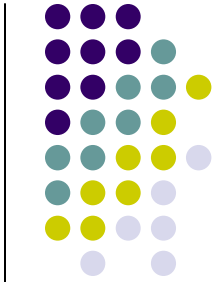
Ext3 Filesystem

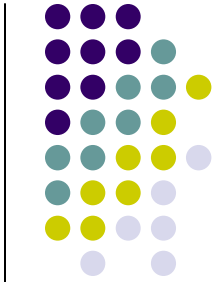


Introduction



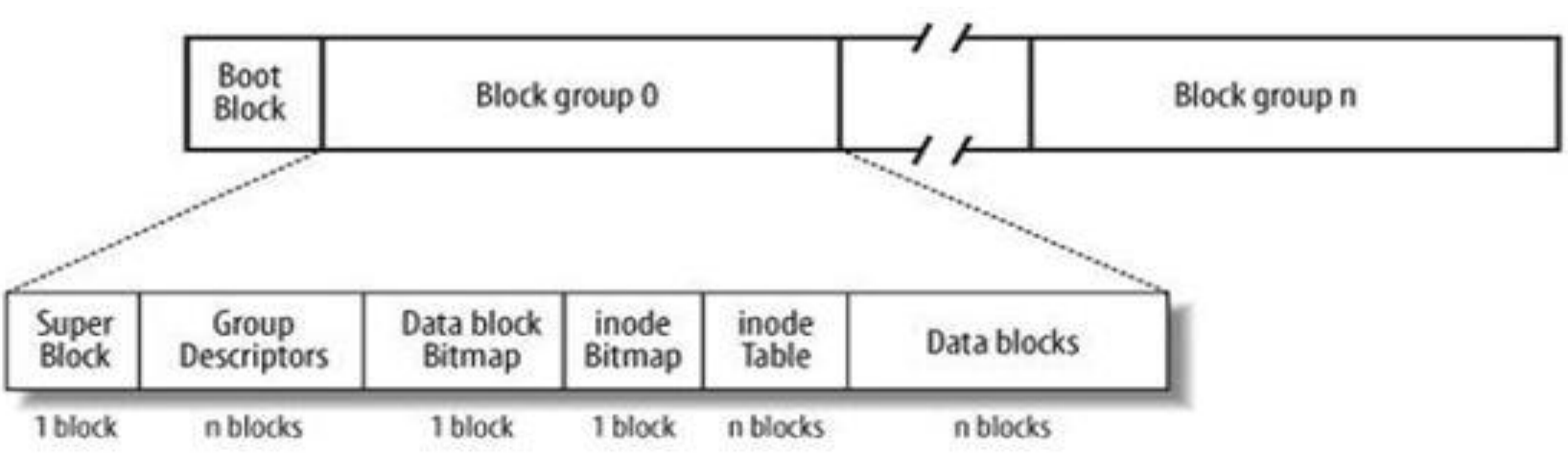
- Common file system on linx
- Introduced in 2001
- Supports max file size of 16 GB to 2 TB
- Max filesystem size can be from 2 TB to 32 TB
- Maximum 32000 sbdirectories under a directory
- Options for block size from 1 KB to 4 KB

Block Groups

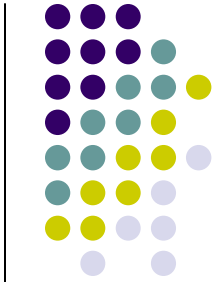


- Disk is partitioned into equal sized block groups
  - Same number of inodes per block group
  - Same number of data blocks per block group
- Each block group has data blocks and inodes stored in adjacent tracks
  - Flies allocated within a single block group usually
  - Inodes and data blocks are close together
  - Reduces average seek time

Partition Layout

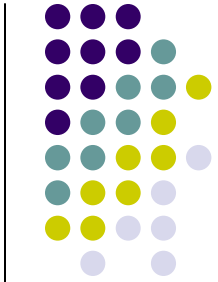


Superblock



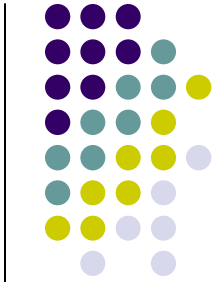
- located 1024 bytes from the start of the file system and is 1024 bytes in size.
- Back up copies are typically stored in the first file data block of each block group
  - Only the one in block group 0 is looked at usually
- Contains a description of the basic size and shape of this file system.

Some Superblock Fields



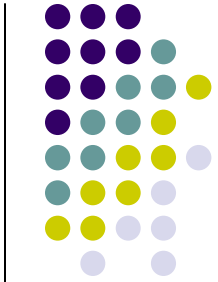
- Block group no. of group storing the superblock
- Block size
- Total no. of blocks
- No. of free blocks
- No. of inodes
- Total no.of free inodes
- First inode (for /)
- Many others, its a long list

The Ext3 Group Descriptor



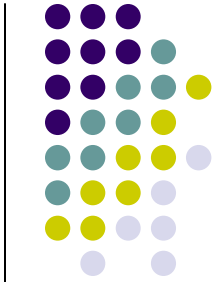
- One Group Descriptor data structure for every block group
- All the group descriptors for all of the Block Groups are duplicated in each Block Group in case of file system corruption.
- The Group Descriptor contains the following:
  - Blocks Bitmap : block number of block allocation bitmap
  - Inode Bitmap : block number of Inode allocation bitmap
  - Inode Table : The block number of the starting block for the Inode table for this Block Group.
  - Free blocks count : number of data blocks free in the Group
  - Free Inodes count : number of Inodes free in the Group
  - Used directory count : number of inodes allocated to directories

Bitmaps



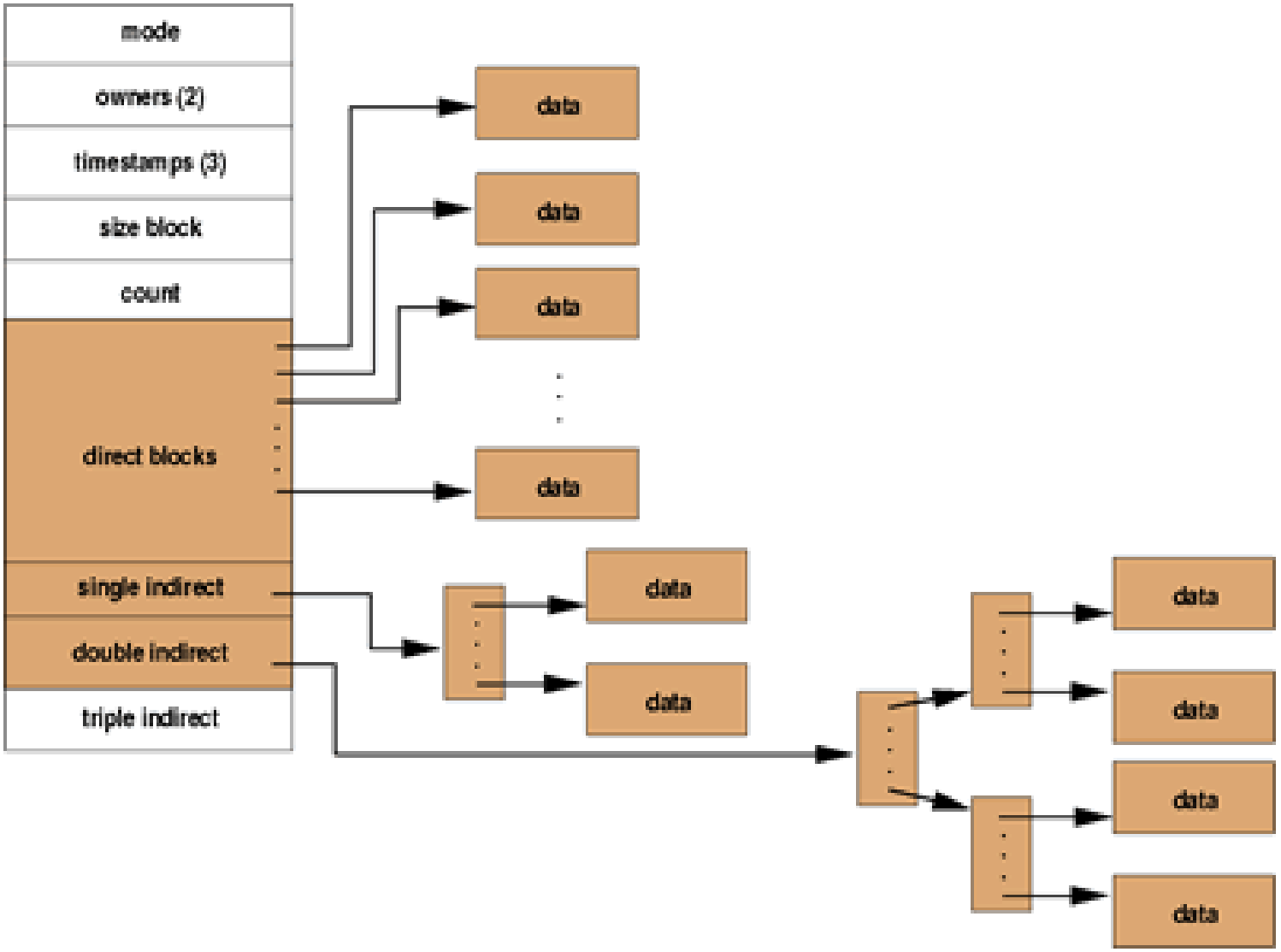
- The block bitmap manages the allocation status of the blocks in the group
- The inode bitmap manages the allocation status of the inodes in the group
- Both bitmaps must fit into one block each
  - Fixes the maximum no. of blocks in a block group as 8 times the block size in bytes

Inodes

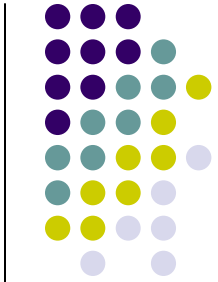


- Inode Tables:
  - Inode table contains the inodes that describe the files in the group
- Inodes:
  - Each inode corresponds to one file, and it stores file's primary metadata, such as file's size, ownership, and temporal information.
  - Inode is typically 128 bytes in size and is allocated to each file and directory
  - 12 direct links, one single, one double, and one triple indirect link

Ext2 inode

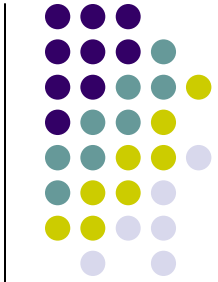

mode
owners (2)
timestamps (3)
size block
count
direct blocks
single indirect
double indirect
triple indirect
data
data
data
data
data
data
data
data
data
data

Some inode fields



- File type
- Access rights
- File length
- Time of last file access
- Time of last change of inode
- Time of last change of file
- Hard links counter
- Number of data blocks
- Pointers to data blocks
- Access control lists

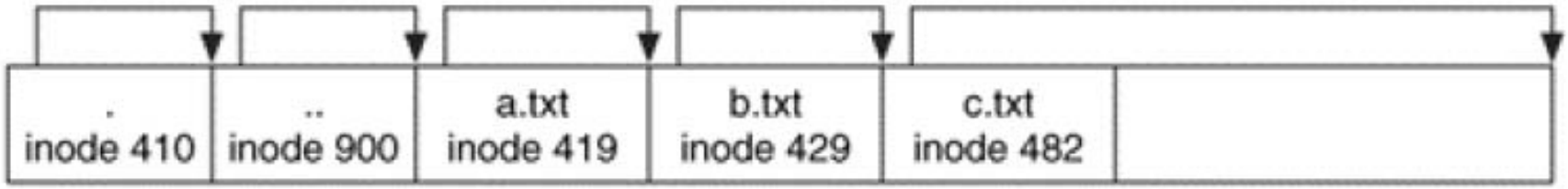
Directories



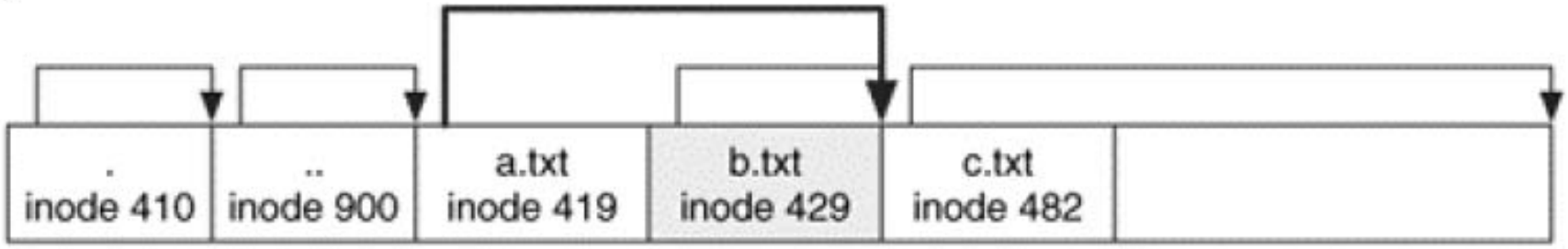
- An Ext3 directory is just like a regular file except it has a special type value.
- The content of directories is a list of directory entry data structure, which describes file/subdirectory name and inode address.
- The length of directory entry varies from 1 to 255 bytes.
- Fields in the directory entry:
  - Inode: inode no. of file/directory
  - Name length: the length of the file name
  - Record length: the length of this directory entry
    - Tells where to start the next structure
  - File type
  - Name: file/subdirectory name

Indexing and Directories

A)

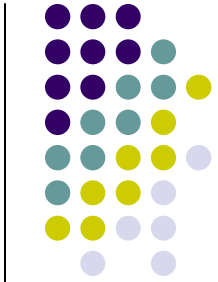


B)

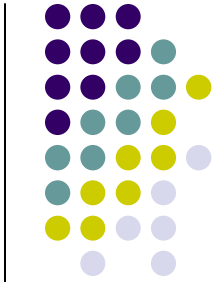


When Ext3 wants to delete a directory entry, it just increases the record length of the previous entry to the end to deleted entry.

Allocating inodes

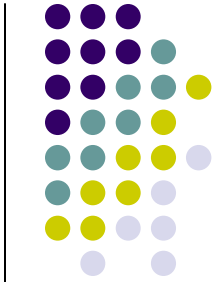


- If a new inode is for a non-directory file, Ext3 allocates an inode in the same block group as the parent directory.
- If that group has no free inode or block,
  - Quadratic search: search in block groups i mod (n), i+1 mod (n), i + 1 + 2 mod (n), i + 1 + 2 + 4 mod (n)
- If quadratic search fails, Ext3 uses exhaustive linear search to find a free inode

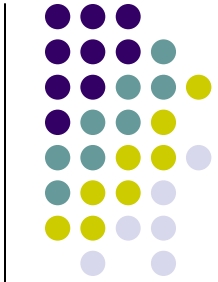


- If a new inode is for a directory, Ext2 tries to place it in a group that has not been used much.
- Using total number of free inodes and blocks in the superblock, Ext3 calculates the average free inodes and blocks per group.
- Ext3 searches each of the group and uses the first one whose free inodes and blocks are less than the average.
- If the pervious search fails, the group with the smallest number of directories is used.

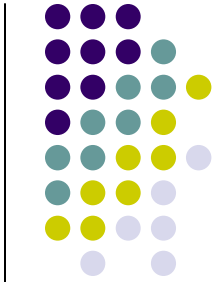
Allocating Data Blocks



- First Goal
  - Get the new block near the last block allocated to the file
- Preallocation – allocates a number of contiguous blocks (usually 8) even if only one block is asked for
  - Preallocated blocks are freed when the file is closed, or when a write operation is not sequential with respect to write operations that triggered the preallocation

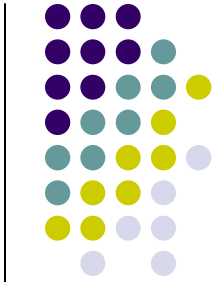


- Every allocation request has a goal block
  - If the current block and previously allocated block have consecutive file block no., goal = logical block no. of previous block + 1
    - Tries to keep consecutive file blocks adjacent on disk
  - Else, if at least one preallocated block earlier, goal = that block
  - Else, goal = first block in the block group with the file's inode
- Aim: allocate a physical block = goal block



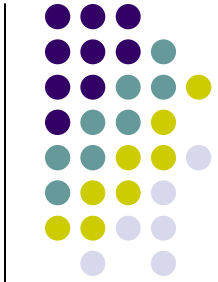
- If the goal block is not free, try the next one
- If not available, search all block groups starting from the one containing the goal block
  - Look for a group of 8 adjacent free blocks
  - If not, look for one

Indexing and Directories

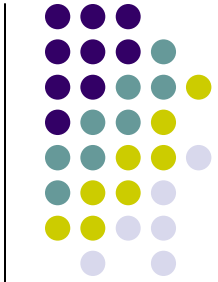


- Directory entry allocation:
  - Ext2 starts at the beginning of the directory and examines each directory entry.
  - Calculate the actual record length and compare with the record length field.
  - If they are different, Ext2 can insert the new directory entry at the end of the current entry.
  - Else, Ext2 appends the new entry to the end of the entry list.

Memory data Structures



- Superblock and Group Descriptors are always cached
- Bitmaps (block and inode) and inode and data blocks are cached dynamically as needed when the corresponding object is in use
  - Page cache mitigates some of the problem of reading from disk



- Main change in Ext3 over Ext2 – add journaling support for recovery
- Not to be discussed today…