TECHNISCHE UNIVERSITÄT DRESDEN

Department of Computer Science, Institute of System Architecture, Operating Systems Group

Distributed Operating Systems

Exercise 1: Robust File Systems

In the tutorial, all solutions will be presented by students. Please be prepared for all questions as the exercise will focus on discussion.

The exercise aims at understanding how file systems work internally and how they can be designed to cope with common failure scenarios during operation of a computer system.

File System Structures

We assume a Unix-like file system that organizes all data and metadata in blocks on the storage medium. Blocks contain either data (D), inodes (I), directory entries (E), an allocation bitmap for inodes or blocks (IA or BA), or the superblock (SB). The size of a block is 4 KiByte.

The following table shows the file system structures of a very simple file system.

Block	0	1	2	3	4	5	6	7	8	9	10
Type	SB	IA	BA	I	E	D	D				
Content	/: 0 free: 4	0-1	0-6	0: 4 1: 5,6	F1: 1	user data	user data				

Inode pointer for root directory “/“

number of free blocks

Inodes 0 and 1 allocated

Blocks 0–6 allocated

Direct block pointers in inodes 0 and 1

Directory entry that maps filename “F1” to inode 1

a) An application opens the file “F1” in the root directory of the file system and reads its contents. Which data and metadata blocks does the operating system have to read from the storage medium?

b) In which order do the accesses take place?

c) What is the role of the buffer cache?

d) The application wants to update the content of the file “F1”. It performs this operation in two steps:

1) The application saves a new version of the file using the filename “F2”, which is 3,000 bytes in size and stored in the same directory.

2) The application deletes the old version named “F1”.

Enter the state of the file system after completing the operations from step 1 and step 2 in the diagram below. Mark all blocks whose content has changed in the respective step.

Block	0	1	2	3	4	5	6	7	8	9	10
Type											
State after step 1											
Content											
changed?											
State after step 2											
Content											
changed?											

Consistency Mechanisms

To increase performance, the operating system keeps new or modified blocks in its buffer cache in main memory. At a later point in time, it writes these blocks to the storage medium in the background.

Consistency-related dependencies between the blocks must be taken into account. We continue to use the same blocks types D, I, N, IA, BA, and SB from the previous task.

e) The storage medium consists of new and innovative *Blitz!* memory cells and is so fast that the “synchronous writes” consistency method can be used with good performance.

Specify a concrete sequence in which the blocks modified in subtask d) can be written to the storage medium. It should be ensured that the file system remains in or can be restored to a consistent state, even if the system crashes before all blocks have been written.

f) Do blocks have to be written multiple times and if so, why?

g) After several weeks of durability tests, it turns out that *Blitz!* memory cells in those blocks that are written to frequently develop defects much earlier than those that are only rarely rewritten.

What type of file system or consistency method would be suitable for extending the life span of the storage medium? Explain how the underlying approach to storage management works and why it can prevent the premature failure of individual memory cells.