Collected here are some remarks regarding FS project description provided in other documents primarily in pdf-file and presentation.

Filesystem Entities

There are several entities that are used in file system design. This project, for instance, uses

- File descriptor
- OFT entry
- Directory entry
- (block usage) Bitmap

With the exception of OFT entries, all of them are stored on disk. To work with entities corresponding types (classes/structures) are introduced in the source code when it is appropriate.

Terminology

Terminology used in this project is a little unconventional. Correspondence between terms used in this project and terms used typically is reflected in following table

This project term	Conventional term
file descriptor	inode (object)
OFT entry	file (object)

Implementation issues

Programming Language

Java, C++, C are acceptable implementation languages. Other propositions *could also be considered*.

Buffering policy

Use of buffers for open files can vary depending on the design decision when to store buffer on disk and when to reload it with data from the other block. One obvious improvement is to add modification bit to the OFT entry so that the buffer is stored only when it was actually changed. But in general, **the choice should be made between two extremes**. In the first approach, buffer updates happen every time the file position “crosses” the block boundaries. Consider if it is possible to avoid redundant block allocation in this case. The second approach is not to update the buffer til the moment the actual data was read from or written to the different block.

Caching

Parts of the file system could be cached in main memory. This includes descriptors, directory entries, bitmap. There are two extremes: a) to keep only structures related to open file in write-through fashion or b) to keep all information and synchronize it only once upon file system initialization and shutdown. **Minimal caching should be used by default** but other options *might be accepted*.

Extensions

Extended configurability

Consider designing a system in such a fashion that file system parameters are not hard-coded but partially entered manually and the rest computed automagically.

File system hierarchy

See item 1 in section 6 of the pdf-file.

Indirect blocks

See item 2 in section 6 of the pdf-file.

Sparse files

Consider extending the system to support sparse files.