Operating Systems
Jacobs University Bremen
Dr. Jürgen Schönwälder

Module: CO-562
Date: 2020-11-12
Due: 2020-11-19

OS 2020 Problem Sheet #10

Problem 10.1: *index node file system* (1+1+1 = 3 points)

You have designed a file system for a block storage device that supports blocks with a size of 128 bytes and blocks are numbered using a 32-bit numbers. Your file system uses index nodes (inodes) and direct, indirect, double indirect, and triple indirect indexes to data blocks. Your file system uses 64 bytes of the inode to store file attributes (file meta data).

a) What is the largest possible file system size? Explain.

b) The inode representing a file stores one indirect, one double indirect, and one triple indirect index, the remaining space is used to store direct indexes to data blocks. All additional index blocks use the entire block to store block numbers. What is the maximum file size? Explain.

c) An application opens a file and seeks to the position 123456. How many blocks (including the inode block) need to be read in order to find the relevant data block? Explain.

Problem 10.2: *find entries in the file system* (2+1+1+1+2 = 7 points)

The `fd` utility you are going to implement in this assignment can be used to easily find files in the file system. The command like syntax is:

```
fd [-t type] [pattern] [path...]
```

If no arguments are given, `fd` will lists all files accessible in the file system tree identified by the current working directory.

```
$ mkdir -p a/b/c/d; touch a/b/x a/b/c/x
$ fd
a
a/b
a/b/c
a/b/c/x
a/b/c/d
a/b/x
```

If a pattern is provided, then the file names (not the path!) are matched against the pattern to determine whether the file path is printed to the standard output. Pattern are interpreted as glob style shell wildcard pattern (see below for an implementation hint).

```
$ fd x
a/b/c/x
a/b/x
$ fd '[a-c]'
a
a/b
a/b/c
```

The `-t` option can be used to filter the search by the file type. The following character is used to select a file type: `f` = regular files, `d` = directories, `l` = symbolic links, `x` = executable files, `e` = empty files, `s` = local domain sockets, `p` = named pipes (FIFOs). The `-t` option can appear multiple times, leading to a logical or of the matching types.

```
$ fd -t f
a/b/x
a/b/c/x
$ fd -t d
a
a/b
a/b/c
a/b/c/d
$ fd -t f -t d
a
a/b
a/b/c
a/b/c/x
a/b/c/d
a/b/x
```

Additional arguments define directories from which to start the search.

```
$ fd 'vm*' /boot
/boot/vmlinuz-4.19.0-12-amd64
/boot/vmlinuz-4.19.0-8-amd64
```

To match a file name against a pattern, you can use the POSIX `fnmatch()` function:

```
#include <fnmatch.h>

int fnmatch(const char *pattern, const char *string, int flags);
```

See the `fnmatch(3)` manual page for a full description.

The programming problem can be broken down in the following smaller steps:

a) Iterate (recursively) through a file system tree.

b) Print file name paths for all accessible files to the standard output.

c) If a pattern is provided on the command line, filter the file names according to the pattern.

d) If search paths are provided on the command line, iterate over all of them (and not over the current working directory).

e) If file type filter are provided (using the `-t` option), filter file names according to the file type filters.