Bilkent University

Department of Computer Engineering

CS 342 - Operating Systems
Project 4 Report

Name: Munib Emre Sevilgen

ID Number: 21602416

Section: 1

Part 1:

The output of the mkfs.ext4 command:

```
------------------------------------------------------------------------------------------------------------------------------
Discarding device blocks: done
Creating filesystem with 49975 4k blocks and 49984 inodes
Filesystem UUID: 9fe55588-6e96-4b2d-bf2c-34e466b3677b
Superblock backups stored on blocks:
        32768
Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
------------------------------------------------------------------------------------------------------------------------------
```

Therefore there are 49984 inodes generated for the file contains 50000 blocks with the 4094 byte block size.

After the virtual disk is unmounted, the files cannot be seen in the mount directory. Then, when we mount again, the files can be seen again.

The output of the dump2fs command:

```
------------------------------------------------------------------------------------------------------------------------------
Filesystem volume name:   <none>
Last mounted on:          /home/emre/Desktop/dir
Filesystem UUID:          9fe55588-6e96-4b2d-bf2c-34e466b3677b
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype extent 64bit flex_bg
sparse_super large_file huge_file dir_nlink extra_isize metadata_csum
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              49984
Block count:              49975
Reserved block count:     2498
Free blocks:              44211
Free inodes:              49965
First block:              0
Block size:               4096
```

```
Fragment size:            4096
Group descriptor size:    64
Reserved GDT blocks:      24
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         24992
Inode blocks per group:   781
Flex block group size:    16
Filesystem created:       Mon May 25 18:21:18 2020
Last mount time:          Mon May 25 18:42:23 2020
Last write time:          Mon May 25 18:45:57 2020
Mount count:              3
Maximum mount count:      -1
Last checked:             Mon May 25 18:21:18 2020
Check interval:           0 (<none>)
Lifetime writes:          170 MB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
Journal inode:            8
Default directory hash:   half_md4
Directory Hash Seed:      80e7d1e2-13db-46c5-bcf5-732e4d7254aa
Journal backup:           inode blocks
Checksum type:            crc32c
Checksum:                 0x79464a27
Journal features:         journal_incompat_revoke journal_64bit journal_checksum_v3
Journal size:             16M
Journal length:           4096
Journal sequence:         0x00000010
Journal start:            0
Journal checksum type:    crc32c
Journal checksum:         0x28924828


Group 0: (Blocks 0-32767) csum 0xdf6a [ITABLE_ZEROED]
  Primary superblock at 0, Group descriptors at 1-1
  Reserved GDT blocks at 2-25
  Block bitmap at 26 (+26), csum 0x6904a36f
  Inode bitmap at 28 (+28), csum 0xb9f4fbd5
  Inode table at 30-810 (+30)
  27078 free blocks, 24973 free inodes, 1 directories, 24970 unused inodes
  Free blocks: 1593-1596, 5694-32767
  Free inodes: 20-24992
Group 1: (Blocks 32768-49974) csum 0x51ec [INODE_UNINIT, ITABLE_ZEROED]
  Backup superblock at 32768, Group descriptors at 32769-32769
  Reserved GDT blocks at 32770-32793
  Block bitmap at 27 (bg #0 + 27), csum 0x6ff1a070
  Inode bitmap at 29 (bg #0 + 29), csum 0x00000000
  Inode table at 811-1591 (bg #0 + 811)
  17133 free blocks, 24992 free inodes, 0 directories, 24992 unused inodes
  Free blocks: 32802-49919, 49956-49967, 49972-49974
  Free inodes: 24993-49984
----------------------------------------------------------------------------------------------------------------------
```

The 44211 blocks are free. There are 2 groups in the file system. In group 0, the bitmap is at the 26th block. The block size is 4096 bytes that are 32768 bits. Therefore the bitmap can contain a bit for each block in the group so, it is big enough. The inode bitmap is at the 28th block. One block for the inode bitmap of group 0 and one for group 1 are occupied. The inode table is at the 30th to 810th blocks in group 0. The 781 blocks from 30th to 810th are occupied for the inode table of group 0. There are 27078 free blocks in group 0.

Part 2:

The sample output of the p2:

```
----------------------------------------------------------------------------------------------------------------------
Name: p1.c
Inode number: 2887942
Type: 8
Number of blocks: 8
Size in bytes: 413
Userid: 1000
---------
Name: p3.c
Inode number: 2884288
Type: 8
Number of blocks: 8
Size in bytes: 1104
Userid: 1000
---------
Name: p2.c
Inode number: 2884282
Type: 8
Number of blocks: 8
Size in bytes: 1290
Userid: 1000
```

```
---------
Name: p1
Inode number: 2883754
Type: 8
Number of blocks: 24
Size in bytes: 8560
Userid: 1000
---------
Name: Makefile
Inode number: 2883752
Type: 8
Number of blocks: 8
Size in bytes: 92
Userid: 1000
---------
Name: p3
Inode number: 2884286
Type: 8
Number of blocks: 32
Size in bytes: 12976
Userid: 1000
---------
Name: p2
Inode number: 2884280
Type: 8
Number of blocks: 32
Size in bytes: 12896
Userid: 1000
---------
Name: project4.pdf
Inode number: 2887422
Type: 8
Number of blocks: 256
Size in bytes: 128542
Userid: 1000
---------
Name: file
Inode number: 2889458
Type: 8
Number of blocks: 399808
Size in bytes: 204700000
Userid: 1000
```

--

Part 3:

The output of the p3 program before the reboot and clearing the cache:
0 seconds and 3 microseconds

The output of the p3 program after the reboot
0 seconds and 62 microseconds

The output of the p3 program after clearing the cache:
0 seconds and 58 microseconds

Therefore the disk cache can decrease the random access times at very high rates. Rebooting the machine also clears the disk cache as expected. When we reboot the machine or clear the disk cache manually, the average random access time increase to 20 times the first output.

Source Code of the p1:

```
--------------------------------------------------------------------------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

int main(int argc, char const *argv[])
{
        if (argc != 2)
        {
                printf("Inputs should be in the following format: p1 N\n");
                exit(0);
        }

        int N = atoi(argv[1]);

        int fd = open("./file", O_RDWR | O_CREAT, 0777);
        char zero = 0;
        int iterNo = 4094 * N;
        for (int i = 0; i < iterNo; ++i)
                write(fd, &zero, 1);
        close(fd);
        return 0;
}
--------------------------------------------------------------------------------------------------------------------------
```

Source Code of the p2:

--

```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/stat.h>
#include <string.h>

int main(int argc, char const *argv[])
{
    if (argc != 2)
    {
        printf("Inputs should be in the following format: p2 P\n");
        exit(0);
    }

    char path[1000];
    strcpy(path, argv[1]);

    DIR *dir = opendir(path);

    if (dir == NULL)
    {
        printf("Unable to read directory");
        exit(0);
    }

    struct dirent *entry;
    struct stat stats;

    while (entry = readdir(dir))
    {
        if (strcmp(".", entry->d_name) == 0 || strcmp("..", entry->d_name) == 0)
            continue;

        printf("---------\n");
        printf("Name: %s \n", entry->d_name);

        char entryPath[1000];
        strcpy(entryPath, path);
        strcat(entryPath, entry->d_name);
        if (stat(entryPath, &stats) == 0)
        {
            printf("Inode number: %ld \n", stats.st_ino);
            printf("Type: %d \n", entry->d_type);
            printf("Number of blocks: %ld \n", stats.st_blocks);
            printf("Size in bytes: %ld \n", stats.st_size);
            printf("Userid: %d \n", stats.st_uid);
        }
        else
        {
            printf("Type: %d - ", entry->d_type);
        }
    }

    closedir(dir);
    return 0;
}
```

Source Code of the p3:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/time.h>
#include <time.h>
#include <string.h>
#include <sys/stat.h>

int main(int argc, char const *argv[])
{
    if (argc != 3)
    {
        printf("Inputs should be in the following format: p3 K F\n");
        exit(0);
    }

    int K = atoi(argv[1]);
    char F[1000];
    strcpy(F, argv[2]);
    int noOfRandomAccess = 1000;

    struct timeval tvStart;
    struct timeval tvEnd;
    struct stat stats;
    long randomPos;

    int fd = open(F, O_RDONLY);
    stat(F, &stats);
    long max = stats.st_size;
    char buf[K + 1];

    gettimeofday(&tvStart, NULL);

    for (int i = 0; i < noOfRandomAccess; i++)
    {
        randomPos = rand() % (max + 1 - 0);
        lseek(fd, randomPos, SEEK_SET);
        read(fd, buf, K);
    }

    gettimeofday(&tvEnd, NULL);

    time_t sec = (tvEnd.tv_sec - tvStart.tv_sec) / noOfRandomAccess;
    suseconds_t usec = (tvEnd.tv_usec - tvStart.tv_usec) / noOfRandomAccess;

    printf("%ld seconds and %ld microseconds\n", sec, usec);
    close(fd);

    return 0;
}
```
