BURAK KORKMAZ
21601296

CS 342 OPERATING SYSTEMS PROJECT 4

At the beginning of this report I want to mention that, I did this project alone. At the beginning of the project, I decided to work on my virtual machine with its own kernel with the version 4.15.0-50-generic. First of all I started with learning Kernel module programming. Later on I started to develop a Kernel Module to in-memory file structures.

My kernel program name is project-module. To compile it you should write “make”. Later than that, you should write “insmod ./module-project.ko processid = <some process id>”. The program will find the process in the PCB(task structure) with traversing the PCB list. I printed descriptor number, current file position pointer, user’s id, process access mode, name of the file, inode number of the file, file length in bytes, number of blocks allocated to the file, directory of the process, blocks that are cached for the process in the page cache, block number, and count as follows.

```
printk(KERN_INFO "1 - descriptor number= %u  \n",index);
                printk(KERN_INFO "2 - current file position number =
%lli  \n",cur->f_pos);
                printk("3 - user id = %u  \n",cur->f_path.dentry-
>d_inode->i_uid);
                printk(KERN_INFO "4 - process access mode = %i  \n",cur-
>f_mode);
                printk(KERN_INFO "5 - name of the file = %s  \n",cur-
>f_path.dentry->d_iname);
                printk(KERN_INFO "6 - inode number of the file = %li
\n",cur->f_path.dentry->d_inode->i_ino);
                printk(KERN_INFO "7 - file length = %lld bytes
\n",cur->f_path.dentry->d_inode->i_size);
                printk(KERN_INFO "8 - number of blocks allocated to file
= %lu    \n",cur->f_path.dentry->d_inode->i_blocks);

printk(KERN_INFO "9 - name of the current directory of the process = %s
\n",path_res);

printk("10 - blocks that are cached for the process in the page cache =
%i \n",totalpage);

printk("11 - the storage device the block is in = %lu \n",bh->b_count);
```

app.c

The app.c is a test program which creates 5 txt files write them accordingly and read after that. The code of the program is mentioned in below. After the execution of the program, I checked the kernel for the output by typing “dmesg”. The output states all related data with the process.

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

int main(){
    int num;

    FILE *fptr;

    fptr = fopen("test1.txt","w");

    for(int i = 0 ; i<1000000; i++){

        fprintf(fptr, "test\n");

    }
    fclose(fptr);

    fptr = fopen("test2.txt","w");

    for(int i = 0 ; i<1000000; i++){
        fprintf(fptr, "test\n");

    }
    fclose(fptr);

    fptr = fopen("test3.txt","w");

    for(int i = 0 ; i<1000000; i++){
        fprintf(fptr,"test\n");

    }
    fclose(fptr);

    fptr = fopen("test4.txt","w");

    for(int i = 0 ; i<1000000; i++){
        fprintf(fptr, "test\n");

    }
    fclose(fptr);
    sleep(1);
    fptr = fopen("test5.txt","w");
    for(int i = 0 ; i<1000000; i++){
```

```
        fprintf(fptr,"test\n");

    }
    fclose(fptr);

    fptr = fopen("test1.txt","r");

    char a[100];
    for(int i = 0 ; i<1000000; i++){

        fgets(a, 100,fptr);
        printf("%s\n",a);
    }
    fclose(fptr);
    sleep(1);
    fptr = fopen("test2.txt","r");
    char b[100];
    for(int i = 0 ; i<1000000; i++){

        fgets(b, 100,fptr);
        printf("%s\n",b);
    }
    fclose(fptr);
    sleep(1);
    fptr = fopen("test3.txt","r");
    char c[100];
    for(int i = 0 ; i<1000000; i++){

        fgets(c, 100,fptr);
        printf("%s\n",c);
    }
    fclose(fptr);
    sleep(1);
    fptr = fopen("test4.txt","r");
    char d[100];
    for(int i = 0 ; i<1000000; i++){

        fgets(d, 100,fptr);
        printf("%s\n",d);
    }
    fclose(fptr);
    sleep(1);
    fptr = fopen("test5.txt","r");
    char e[100];
    for(int i = 0 ; i<1000000; i++){

        fgets(e, 100,fptr);
        printf("%s\n",e);
    }


    fclose(fptr);

}
```

The output of the app.c in a Kernel log file

[11012.487352] Hello World
[11012.487391] PCB of the process with given pid = 7454
[11012.487392] 1 - descriptor number= 0
[11012.487393] 2 - current file position number = 0
[11012.487394] 3 - user id = 1000
[11012.487395] 4 - process access mode = 393219
[11012.487395] 5 - name of the file = 0
[11012.487396] 6 - inode number of the file = 3
[11012.487397] 7 - file length = 0 bytes
[11012.487397] 8 - number of blocks allocated to file = 0
[11012.487398] 1 - descriptor number= 1
[11012.487398] 2 - current file position number = 0
[11012.487399] 3 - user id = 1000
[11012.487400] 4 - process access mode = 393219
[11012.487400] 5 - name of the file = 0
[11012.487401] 6 - inode number of the file = 3
[11012.487401] 7 - file length = 0 bytes
[11012.487402] 8 - number of blocks allocated to file = 0
[11012.487402] 1 - descriptor number= 2
[11012.487403] 2 - current file position number = 0
[11012.487403] 3 - user id = 1000
[11012.487404] 4 - process access mode = 393219
[11012.487404] 5 - name of the file = 0
[11012.487405] 6 - inode number of the file = 3
[11012.487405] 7 - file length = 0 bytes
[11012.487406] 8 - number of blocks allocated to file = 0
[11012.487406] 1 - descriptor number= 3
[11012.487407] 2 - current file position number = 0
[11012.487408] 3 - user id = 1000
[11012.487408] 4 - process access mode = 134578206
[11012.487409] 5 - name of the file = test1.txt
[11012.487410] 6 - inode number of the file = 786885
[11012.487410] 7 - file length = 5000000 bytes
[11012.487411] 8 - number of blocks allocated to file = 9768
[11012.487433] 9 - name of the current directory of the process = /home/burak/Desktop/project4/module
[11012.487434] 10 - blocks that are cached for the process in the page cache = 1221