COMP 304: Assignment 3

Doğa Ege İnhanlı – 69033

Problem 1)

- External fragmentation may occur with contiguous allocation and segmentation but it may not occur with paging.
- Code sharing may occur with segmentation and paging, but it may not occur with contiguous allocation.

Problem 2)

- We have 12 + 16 = 28 bits for page table field. The remaining 12 bits are for the offset. 2^12 = 4KB
- In total, 28 bits are for the page table field. That's why, there are 2^28 = 268435456 pages in the address space.

Problem 3:

• FIFO:

Page reference string:

①	②	③	④	2	1	⑤	⑥	2	1	2	3	⑦	6	3	2	①	②	③	6
1	1	1	1			1	1					7				7	7	7	
	2	2	2			2	2					2				1	1	1	
		3	3			3	3					3				3	2	2	
			4			4	4					4				4	4	3	
						5	5					5				5	5	5	
							6					6				6	6	6	

In total, there are $\underline{\underline{10}}$ page faults.

• Optimal

①	②	③	④	2	1	⑤	⑥	2	1	2	3	⑦	6	3	2	1	2	3	6
1	1	1	1			1	1					1							
	2	2	2			2	2					2							
		3	3			3	3					3							
			4			4	4					7							
						5	5					5							
							6					6							

In total, there are $\underline{\underline{7}}$ page faults.

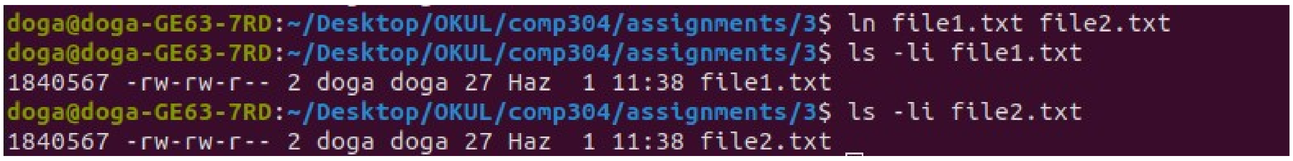
• LRU:

①	②	③	④	2	1	⑤	⑥	2	1	2	3	⑦	6	3	2	1	2	3	6
1	1	1	1			1	1					1							
	2	2	2			2	2					2							
		3	3			3	3					3							
			4			4	4					7							
						5	5					5							
							6					6							

In total, there are $\underline{\underline{7}}$ page faults.

Problem 4)

a) Inode values are the same. The contents are the same, too.

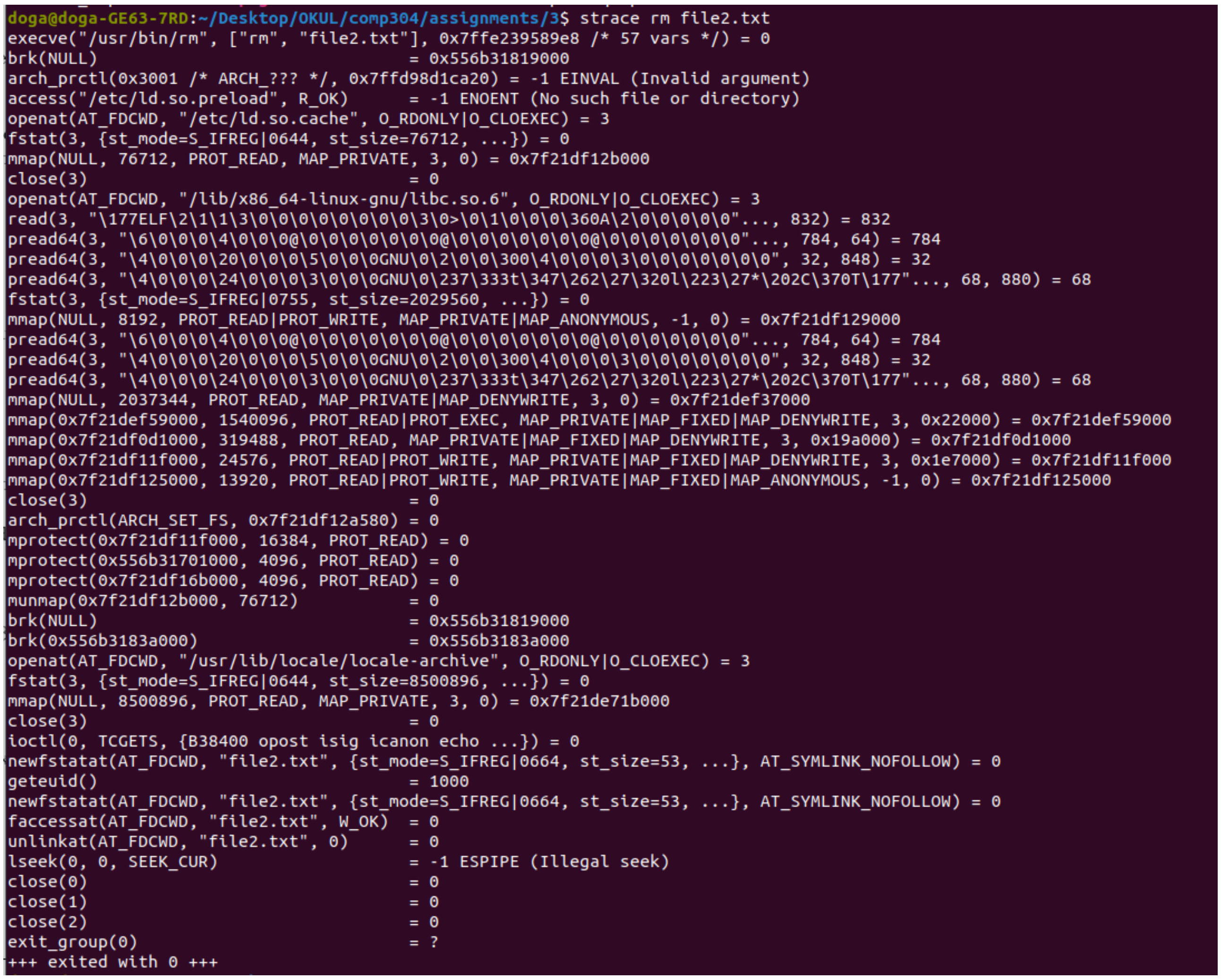


```
doga@doga-GE63-7RD:~/Desktop/OKUL/comp304/assignments/3$ ln file1.txt file2.txt
doga@doga-GE63-7RD:~/Desktop/OKUL/comp304/assignments/3$ ls -li file1.txt
1840567 -rw-rw-r-- 2 doga doga 27 Haz  1 11:38 file1.txt
doga@doga-GE63-7RD:~/Desktop/OKUL/comp304/assignments/3$ ls -li file2.txt
1840567 -rw-rw-r-- 2 doga doga 27 Haz  1 11:38 file2.txt
```

b) After editing the content of the file2.txt, the content of the file1.txt became the same with the content of the file2.txt. Inode values are still the same.

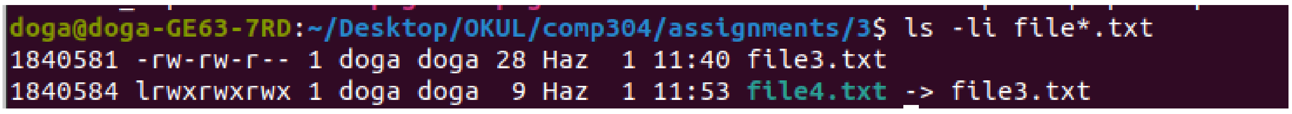
c) Yes, file2.txt still exists.

d)



```
doga@doga-GE63-7RD:~/Desktop/OKUL/comp304/assignments/3$ strace rm file2.txt
execve("/usr/bin/rm", ["rm", "file2.txt"], 0x7ffe239589e8 /* 57 vars */) = 0
brk(NULL)                               = 0x556b31819000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd98d1ca20) = -1 EINVAL (Invalid argument)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=76712, ...}) = 0
mmap(NULL, 76712, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f21df12b000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360A\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\237\333t\347\262\27\320l\223\27*\202C\370T\177"..., 68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029560, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f21df129000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\237\333t\347\262\27\320l\223\27*\202C\370T\177"..., 68, 880) = 68
mmap(NULL, 2037344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f21def37000
mmap(0x7f21def59000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) = 0x7f21def59000
mmap(0x7f21df0d1000, 319488, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19a000) = 0x7f21df0d1000
mmap(0x7f21df11f000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f21df11f000
mmap(0x7f21df125000, 13920, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f21df125000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f21df12a580) = 0
mprotect(0x7f21df11f000, 16384, PROT_READ) = 0
mprotect(0x556b31701000, 4096, PROT_READ) = 0
mprotect(0x7f21df16b000, 4096, PROT_READ) = 0
munmap(0x7f21df12b000, 76712)           = 0
brk(NULL)                               = 0x556b31819000
brk(0x556b3183a000)                     = 0x556b3183a000
openat(AT_FDCWD, "/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=8500896, ...}) = 0
mmap(NULL, 8500896, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f21de71b000
close(3)                                = 0
ioctl(0, TCGETS, {B38400 opost isig icanon echo ...}) = 0
newfstatat(AT_FDCWD, "file2.txt", {st_mode=S_IFREG|0664, st_size=53, ...}, AT_SYMLINK_NOFOLLOW) = 0
geteuid()                               = 1000
newfstatat(AT_FDCWD, "file2.txt", {st_mode=S_IFREG|0664, st_size=53, ...}, AT_SYMLINK_NOFOLLOW) = 0
faccessat(AT_FDCWD, "file2.txt", W_OK)  = 0
unlinkat(AT_FDCWD, "file2.txt", 0)      = 0
lseek(0, 0, SEEK_CUR)                   = -1 ESPIPE (Illegal seek)
close(0)                                = 0
close(1)                                = 0
close(2)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

e) The inode values are unique for each.


```
doga@doga-GE63-7RD:~/Desktop/OKUL/comp304/assignments/3$ ls -li file*.txt
1840581 -rw-rw-r-- 1 doga doga 28 Haz  1 11:40 file3.txt
1840584 lrwxrwxrwx 1 doga doga  9 Haz  1 11:53 file4.txt -> file3.txt
```

f) Yes, it has been altered, too.

g) After deleting file3.txt, I observed that the content of the file4.txt was gone. When I tried to write something to file4.txt, I observed that it created a copy of file4.txt with name file3.txt.

h) Hard links link the files with the same inode values and same permissions. On the other hand, soft links link the files with distinct inode values and distinct permissions. Another difference is that the original file is deleted, the hard link is not deleted because the content does not change. Nevertheless, the soft link is deleted because the content is reset in my experiment. Hard link are used only for some jobs which needs root access. Soft links are used commonly.