ANOTAÇÕES

- baixou linux kernel 4.8
- pasta fs/minix
- copiou pasta minix para “pasta pessoal”
- terminal comandos de **https://www.ibm.com/developerworks/library/l-linux-filesystem/index.html**

```
$ dd if=/dev/zero of=file.img bs=1k count=10000
$ losetup /dev/loop0 file.img
$ mkfs.minix -c /dev/loop0 10000
```

to be continued……

pretenciosos garotos querem:

- compilar modulo novo com printk. **ok**
- tirar o minix.ko e subir o novo. **ok**
- executar ações que utilizem esse módulo
- ver se printk aparecem no log **ok**

por hoje eh soh

só salvando esses links aqui (ignorar):
https://minix1.woodhull.com/faq/mxfromlx.html

24/10

Para aparecer o *Volume 10MB*:
> losetup /dev/loop0 file.img

Desmontar/remover partição (apenas quando não utilizando):
> umount /media/gruposob/disk
> rmmod minix.ko

Lembrete: Ao clicar em *Volume 10MB* ele faz um mount sozinho, então é impossível removê-lo com o *rmmod*. É necessário fechá-lo antes.

Vamos no *inode.c* e no seu *init_minix_fs* colocamos um printk para teste de depuração.
No *Makefile* modificamos para *"(...) := minix_1.o"*.

> make
> insmod minix_1.ko
> dmesg

Nosso printk apareceu. Great Success. Parabéns. Muito bem, Paulinho.

Criamos um arquivo .txt para teste, para ver se realmente ele está sendo utilizado:
> cd /media/gruposob/disk
> nano teste.txt
> rmmod minix_1

Está sendo utilizado, como previámos. Confirmando nossa teoria. Removemos este arquivo *teste*.

Vamos em *dir.c* em *minix_make_empty* colocamos um printk para depuração.
Testamos:
> umount /media/gruposob/disk
> rmmod minix_1.ko
> make
> insmod minix_1.ko
> mount /media/gruposob/disk (OU SIMPLESMENTE ABRIR *Volume 10MB*)

> mkdir pasta1
> dmesg

Nosso printk apareceu. Great Success. Parabéns. Muito bem, Paulinho. Então entra na função *minix_make_empty* quando é criado algo no diretório.

Removemos a pasta criada:
> rmdir pasta1

Criamos um arquivo para teste:
> nano teste.txt
> dmesg

Nosso printk apareceu. Great Success. Parabéns. Muito bem, Paulinho. Então entra na função *minix_make_empty* quando é criado algo no diretório. Agora ainda mais.

Sempre que criamos pastas ou arquivos entra nessa função.

Vamos em *bitmap.c* em *minix_new_block* colocamos um printk para depuração.
Testamos:
> umount /media/gruposob/disk
> rmmod minix_1.ko
> make
> insmod minix_1.ko
> mount /media/gruposob/disk (OU SIMPLESMENTE ABRIR *Volume 10MB*)

Criamos uma pasta no diretório.
> dmesg
Nosso printk apareceu. Great Success. Parabéns. Muito bem, Paulinho.

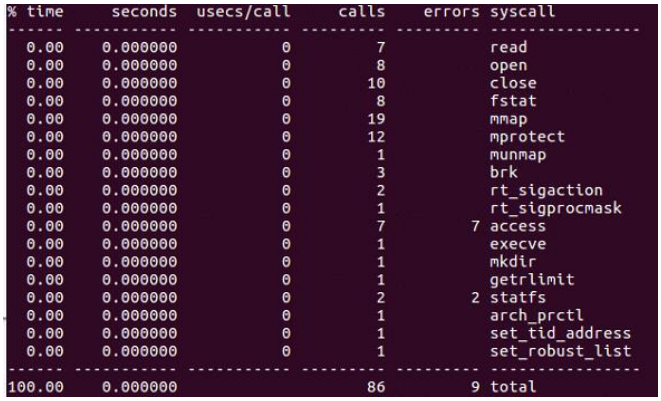
Criamos um arquivo.txt na pasta.
Cada vez que criamos, escrevemos no arquivo e salvamos ou deletamos, um printk aparece.

Vamos em *namei.c* em *minix_mkdir* e vemos que é ela quem chama as outras funções como:

- *minix_new_inode*
- *minix_set_inode*
- *minix_inc_link_count*
- *minix_make_empty*

Teste com strace
Dentro do /media/gruposob/disk:

> strace -c mkdir teste



% time	seconds	usecs/call	calls	errors	syscall
0.00	0.000000	0	7		read
0.00	0.000000	0	8		open
0.00	0.000000	0	10		close
0.00	0.000000	0	8		fstat
0.00	0.000000	0	19		mmap
0.00	0.000000	0	12		mprotect
0.00	0.000000	0	1		munmap
0.00	0.000000	0	3		brk
0.00	0.000000	0	2		rt_sigaction
0.00	0.000000	0	1		rt_sigprocmask
0.00	0.000000	0	7	7	access
0.00	0.000000	0	1		execve
0.00	0.000000	0	1		mkdir
0.00	0.000000	0	1		getrlimit
0.00	0.000000	0	2	2	statfs
0.00	0.000000	0	1		arch_prctl
0.00	0.000000	0	1		set_tid_address
0.00	0.000000	0	1		set_robust_list
100.00	0.000000		86	9	total

> strace -e open mkdir teste2

```
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
open("/lib/x86_64-linux-gnu/libselinux.so.1", O_RDONLY|O_CLOEXEC) = 3
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
open("/lib/x86_64-linux-gnu/libpcre.so.3", O_RDONLY|O_CLOEXEC) = 3
open("/lib/x86_64-linux-gnu/libdl.so.2", O_RDONLY|O_CLOEXEC) = 3
open("/lib/x86_64-linux-gnu/libpthread.so.0", O_RDONLY|O_CLOEXEC) = 3
open("/proc/filesystems", O_RDONLY)     = 3
open("/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
+++ exited with 0 +++
```

Outro dia

No arquivo

Em *file.c* temos a operação *.write_iter = generic_file_write_iter*. Que todo arquivo escrito passará por ela, é nela que trabalharemos.
Nela recebemos uma struct do tipo file (que seria o nosso arquivo), e um iov_iter.

```
ssize_t generic_file_write_iter(struct kiocb *iocb, struct iov_iter *from)
{
        struct file *file = iocb->ki_filp;
        struct inode *inode = file->f_mapping->host;
        ssize_t ret;

        mutex_lock(&inode->i_mutex);
        ret = generic_write_checks(iocb, from);
        if (ret > 0)
                ret = __generic_file_write_iter(iocb, from);
        mutex_unlock(&inode->i_mutex);

        if (ret > 0) {
                ssize_t err;

                err = generic_write_sync(file, iocb->ki_pos - ret, ret);
                if (err < 0)
                        ret = err;
        }
        return ret;
}
EXPORT_SYMBOL(generic_file_write_iter);


struct kiocb {
        struct file             *ki_filp;
        loff_t                  ki_pos;
        void (*ki_complete)(struct kiocb *iocb, long ret, long ret2);
        void                    *private;
        int                     ki_flags;
        enum rw_hint            ki_hint;
} __randomize_layout;


struct iov_iter {
        int type;
        size_t iov_offset;
        size_t count;
        union {
                const struct iovec *iov;
                const struct kvec *kvec;
                const struct bio_vec *bvec;
                struct pipe_inode_info *pipe;
        };
        union {
                unsigned long nr_segs;
                struct {
                        int idx;
                        int start_idx;
                };
        };
};
```

A ideia é remover esse generic_file_write_iter para colocar a nossa função nova.
Para isso precisamos criar uma função ssize_t;
Para isso precisamos do UNISTD.H.

Tentativas:
sudo apt-get install libc6-dev libncursesw5-dev
sudo apt-get install build-essential libncursesw5-dev

OUTRA APROXIMAÇÃO PARA O PROBLEMA:

Criamos funções na nossa file.c que tem um printk, e chamar as antigas funções dos file_operations dentro dessas funções novas.