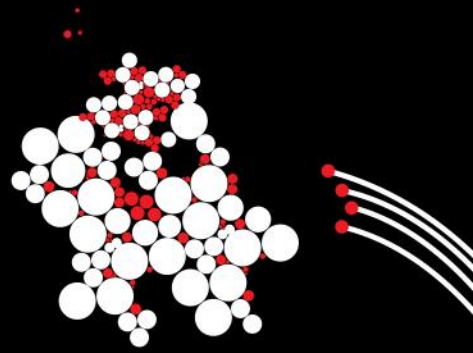
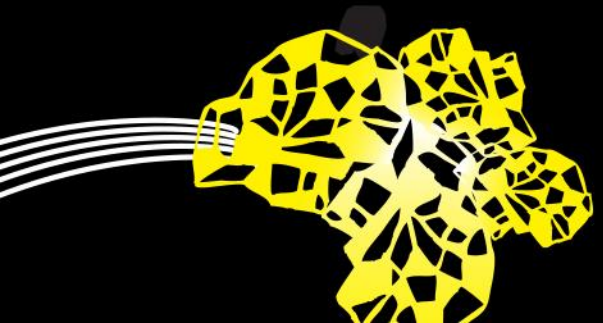
UNIVERSITY OF TWENTE.



OPERATING SYSTEMS

FILESYSTEMS

ERIK TEWS <e.tews@utwente.nl>



FILESYSTEMS

- Harddisks/SSDs and similar devices just provide a block device
  - A long sequential list of blocks/bytes
- We would like to store data in a more convenient way
  - We call that files which are usually organized in directories
- A filesystem provides more or less a way how we can interpret that long list of bytes/blocks as files and directories

DIFFERENT FILESYSTEMS

- There is more than one way to solve that problem
  - Each one has some advantages and disadvantages
- Linux often uses the EXT4 filesystem
  - Based on EXT3, based on EXT2….
- We will focus here mostly on EXT2

LINUX FILESYSTEM SUPPORT

- EXT2
  - Traditional old Linux filesystem
- EXT3/4
  - Further development of ext2
  - Journaling, efficiency
- Btrfs, ZFS
  - Provide more features
  - Partially also implement RAID

OTHER FILE SYSTEMS

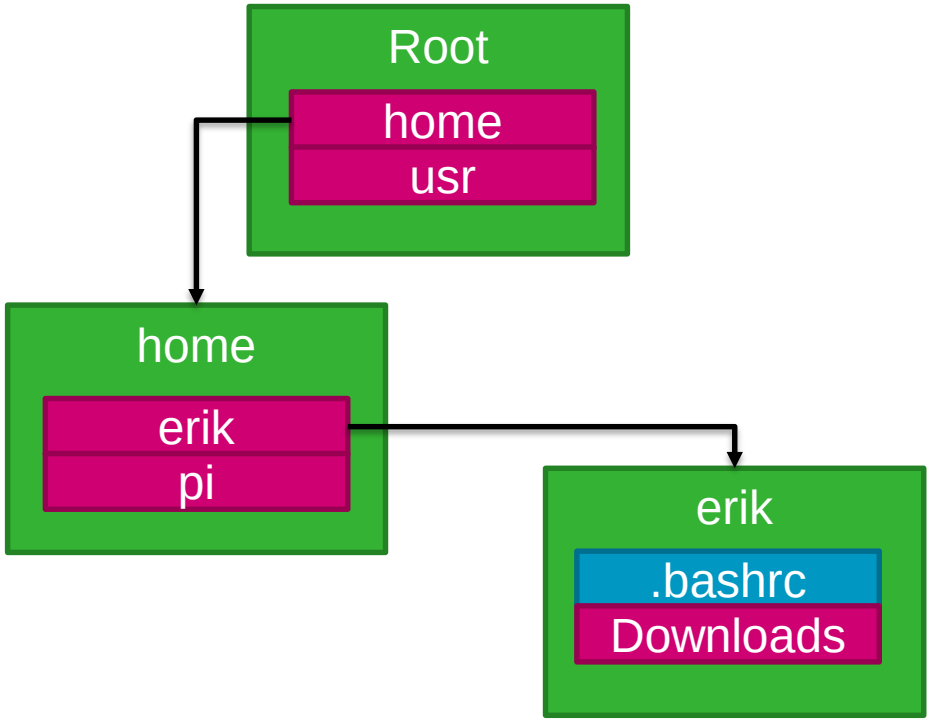
- Windows uses NTFS
  - Older versions and simple devices use (V)FAT
  - Partially there is also EXFAT as well
- Linux also has virtual file system
  - proc, sys…

FILES

- Properties
  - Long term existence of data
  - Sharable between processes
  - Access control
- Operations
  - Create/Delete
  - Open/Close
  - Read/Write/Seek
- Structure
  - Bytes/Records

DIRECTORIES

- Path name
  - Current directory
  - Parent directory
  - Root directory
- ls -ld . .. /

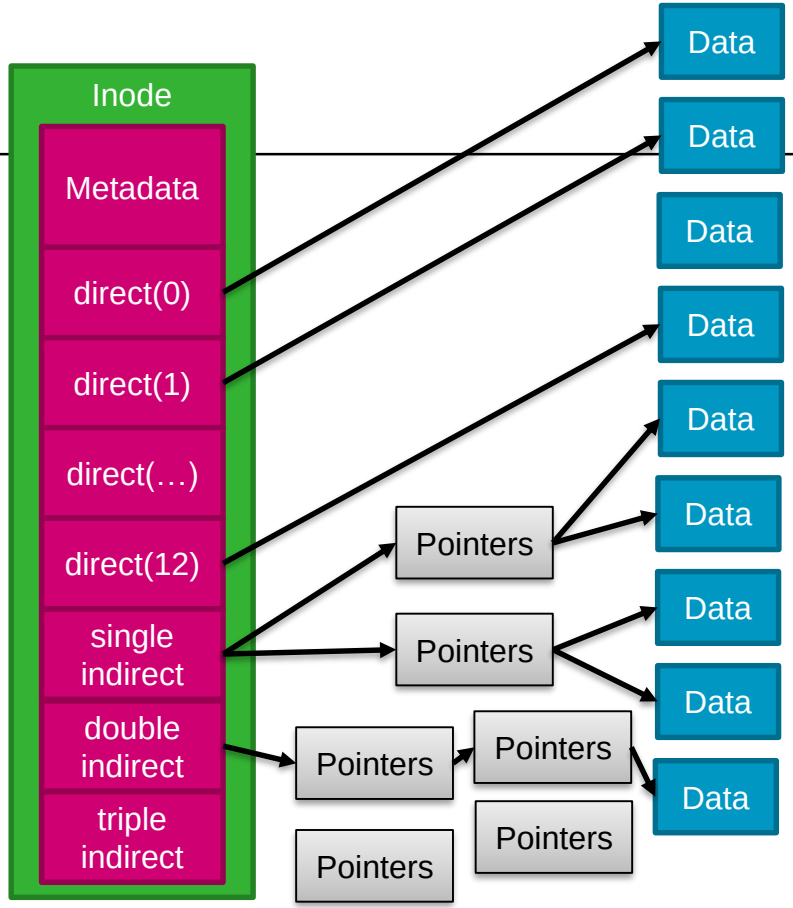


LINUX EXT2 PARTITION

- Per device:
  - Boot block
  - Super block
  - Inode table
  - Data blocks

- stat -f /
- stat -f $HOME

- ls -ldi . .. ../.. ../../.. / /..



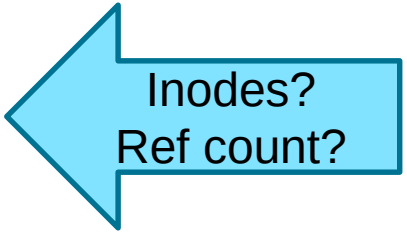
INODE DATA

DEBUGFS /DEV/MMCBLK0P2 STAT <141978>

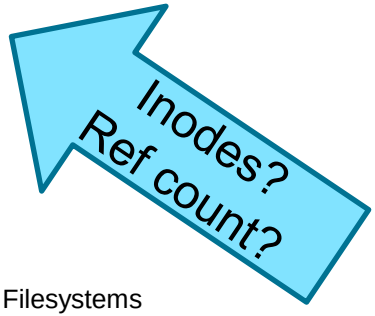
```
Inode: 141978   Type: regular    Mode:  0644   Flags: 0x80000
Generation: 1659629937    Version: 0x00000000:00000001
User:  1000   Group:  1000   Project:     0   Size: 1080
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 8
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x5bb31516:82eb4120 -- Tue Oct  2 06:49:58 2018
 atime: 0x5bb10687:00000000 -- Sun Sep 30 17:23:19 2018
 mtime: 0x5bb10687:00000000 -- Sun Sep 30 17:23:19 2018
crtime: 0x5bb31516:82eb4120 -- Tue Oct  2 06:49:58 2018
Size of extra inode fields: 32
EXTENTS:
(0):606238
```

HARD AND SYMBOLIC LINKS

- rm a b c
- echo "Hello World" >a
- ln a b
- ls -li a b
- rm a
- cat b
- ln b a
- ls -li a b

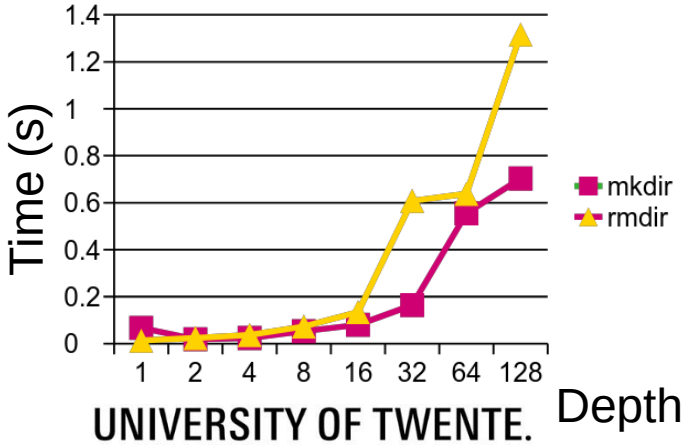


- ln -s a c
- cat c
- rm a b
- ls -li c
- cat c
- echo "Hello World" >a
- ls -li a c
- cat c



MAKING AND REMOVING A DIRECTORY

- gcc Mkdir.c
- ./a.out 4
- ls –lRi Foo*
- strace ./a.out 4




```c
int main(int argc, char* argv[]) {
    int i, n = (argc==1?1:atoi(argv[1]));
    char top[M], cur[M], tmp[M];
    getcwd(top,M);
    printf("top=%s\n",top);
    strcpy(cur, ".");
    for(i=0; i<n; i++) {
        strcpy(tmp, cur);
        sprintf(cur, "%s/Foo", tmp);
        mkdir(cur, 0755);
        sprintf(tmp, "%s/Foo_%d", top, i);
        symlink(cur, tmp);
    }
    return 0;
}
```

READING A DIRECTORY

- gcc Readdir.c
- ./a.out /
- ./a.out .

- Why is the root inode #2

```
int main(int argc, char * argv[]) {
    DIR *dirp = opendir(argv[1]) ;
    for(;;) {
        struct dirent *dp = readdir(dirp);
        if( dp != NULL ) {
            char t = … ;
            printf("%d %c %s\n",
                dp->d_ino, t,
                dp->d_name);
        } else {
            break ;
        }
    }
    closedir(dirp);
    return 0;
}
```

UNNAMED PIPE

- Output?
- gcc Pipe.c
- ./a.out

- Try this a few times…

```
int main(int argc, char **argv) {
    int r, fd[2];
    pipe(fd);
    pid_t pid=fork();
    if(pid== 0) {
        close(fd[1]);
        read(fd[0], &r, sizeof (int));
        printf("child  %d\n", r);
        close(fd[0]);
    } else {
        close(fd[0]);
        srand(time(NULL));
        r = rand();
        printf("parent %d\n", r);
        write(fd[1], &r, sizeof (int));
        close(fd[1]);
        waitpid(pid,0,0);
    }
    return 0;
}
```

NAMED PIPE

- Output?
- gcc Fifo.c
- ./a.out foo
- ls –l foo

```
int main(int argc, char **argv) {
    int r;
    mkfifo(argv[1], 0666) ;
    pid_t pid=fork();
    if(pid== 0) {
        int fd = open(argv[1], O_RDONLY);
        read(fd, &r, sizeof (int));
        printf("child  %d\n", r);
        close(fd);
    } else {
        int fd = open(argv[1], O_WRONLY, 0666);
        srand(time(NULL));
        r = rand();
        printf("parent %d\n", r);
        write(fd, &r, sizeof (int));
        close(fd);
        waitpid(pid,0,0);
    }
    return 0;
}
```

MOUNT POINTS

- more /proc/mounts
- gcc Mount.c
- ./a.out $HOME

- lookup(“/home/lecturer”,…)
- lookup(“/home”,…)
- lookup(“/”,…)

```
char *lookup(char *pth, char *mnt) {
    struct mntent m;
    struct stat s;
    stat(pth, &s);
    dev_t d = s.st_dev;
    ino_t i = s.st_ino;
    FILE *f = setmntent("/proc/mounts", "r");
    while (getmntent_r(f, &m, mnt, M)) {
        if (stat(m.mnt_dir, &s) != 0) {
            continue;
        }
        if (s.st_dev == d && s.st_ino == i) {
            return mnt ;
        }
    }
    endmntent(f);
    return NULL;
}
```

SUMMARY

- File management maps logical files onto disk blocks
- Directory structure allows efficient lookup of files
- Disk blocks are used for paging and swapping too
- Everything is cached in memory