

# Lab 5 - Beniamin Jankowski

---

## Zad 1.1

Komendy do utworzenia plików:

```
dd if=/dev/zero of=zero.dat bs=1M count=100
```

```
dd if=/dev/urandom of=losowy.dat bs=1M count=100
```

```
(kali㉿kali)-[~/Desktop]
$ dd if=/dev/zero of=zero.dat bs=1M count=100
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.568698 s, 184 MB/s

(kali㉿kali)-[~/Desktop]
$ dd if=/dev/urandom of=losowy.dat bs=1M count=100
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.979413 s, 107 MB/s
```

Komenda do spakowania:

```
zip arch.zip zero.dat losowy.dat
```

```
(kali㉿kali)-[~/Desktop]
$ ll | grep -E "(.dat)|(.zip)"
-rw-r--r-- 1 kali kali 104976362 Jan  3 18:38 arch.zip
-rw-r--r-- 1 kali kali 104857600 Jan  3 18:31 losowy.dat
-rw-r--r-- 1 kali kali 104857600 Jan  3 18:31 zero.dat
```

Pliki po spakowaniu są mniejsze ponieważ nastąpiła ich kompresja, gdzie w zero.dat kompresja była bardzo skuteczna z racji na powtarzające się puste bajty.

## Zad 1.2

Powiązanie miękkie:

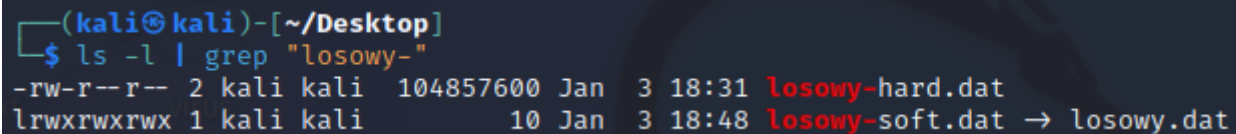
```
ln -s losowy.dat losowy-soft.dat
```

Powiązanie twarde:

```
ln losowy.dat losowy-hard.dat
```

Aby sprawdzić zajętość:

```
ls -l | grep "losowy-"
```



```
(kali㉿kali)-[~/Desktop]
$ ls -l | grep "losowy-"
-rw-r--r-- 2 kali kali 104857600 Jan  3 18:31 losowy-hard.dat
lrwxrwxrwx 1 kali kali      10 Jan  3 18:48 losowy-soft.dat -> losowy.dat
```

Dowiązanie miękkie to wskaźnik na nazwę pliku - jeśli zniknie plik, zniknie również powiązanie. Stąd tak mała waga pliku.

Dowiązanie twarde to tak naprawdę kopia pliku. Nawet jeśli zniknie plik, to dowiązanie twarde zostanie.

## Zad 1.3

Utwórz plik:

```
dd if=/dev/zero of=pusty.dat bs=1M count=100
```

Zamontuj system plików:

```
mkfs.ext4 pusty.dat
```

Utwórz katalogm gdzie chcemy zamontować system plików:

```
mkdir /mnt/mounting
```

Zamontujmy system plików

```
mount -t ext4 pusty.dat /mnt/mounting
```

Następnie należy dodać linię do pliku /etc/fstab:

```
pusty.dat /mnt/mounting ext4 defaults,user 0 0
```

Po powyższej linijce nie musimy używać sudo

## Zad 1.5

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define BUFFER_SIZE 8192

int main(int argc, char *argv[]) {

    int buffer_size = atoi(argv[1]);
    const char *src_file = argv[2];
    const char *target_file = argv[3];

    // Otwieranie plików
    FILE *src = fopen(src_file, "rb");
    FILE *target = fopen(target_file, "wb");

    // Alokowanie bufera
    char *buffer = (char*) malloc(buffer_size);

    // Kopiowanie pliku
    size_t bytes_read;
    while ((bytes_read = fread(buffer, 1, buffer_size, src)) > 0) {
        fwrite(buffer, 1, bytes_read, target);
    }

    free(buffer);
    fclose(src);
    fclose(target);

    return 0;
}
```

## Zad 1.6

```
#!/bin/bash

declare -A visited #tablica asocjacyjna

function check_link {

    local link=$1
    local count=0

    while [ -L "$link" ]
    do
        count=$((count + 1))
        if [ ${visited["$link"]} ];
        then
```

```

        echo "Zapełnienie: $link -> ${visited["$link"]}"
        return
    fi

    visited["$link"]=$(readlink "$link")
    link=${visited["$link"]}
done

echo "Brak zapełnień: $count"
}

for file in "$1"/* #wszystkie pliki i katalogi katalogu $1
do

    if [ -L "$file" ]
    then
        check_link "$file"
    fi

done

```

## Zad 1.7

```

#!/bin/bash

files=$(find "$1" -type f -maxdepth 1)

for file in $files; do
    links=$(find "$1" -samefile "$file" | wc -l)
    if [ "$links" -gt 1 ]
    then
        echo "$file has $links hard links"
    fi
done

```

## Zad 1.8

```
ls -Rl $1 | grep -vE "(^\.)|(^t)|(^s*$)" | sort | cut -d" " -f1 | uniq -c
```

## Zad 1.9

```

dd if=/dev/zero of=disc_512 bs=1M count=100
dd if=/dev/zero of=disc_1024 bs=1M count=100
dd if=/dev/zero of=disc_2048 bs=1M count=100
dd if=/dev/zero of=disc_4096 bs=1M count=100

```

```
mkfs.ext4 -b 512 disc_512 #tutaj będzie najwięcej plików
mkfs.ext4 -b 1024 disc_1024
mkfs.ext4 -b 2048 disc_2048
mkfs.ext4 -b 4096 disc_4096
```

```
losetup -f disc_512
losetup -f disc_1024
losetup -f disc_2048
losetup -f disc_4096
```

```
for i in $(seq 1 1000)
do
    dd if=/dev/zero of=disc_512/$i bs=512 count=1
    dd if=/dev/zero of=disc_1024/$i bs=1024 count=1
    dd if=/dev/zero of=disc_2048/$i bs=2048 count=1
    dd if=/dev/zero of=disc_4096/$i bs=4096 count=1
done
```

```
for i in $(seq 1 1000)
do
    size=$(( ( RANDOM % 8192 ) + 256 ))

    dd if=/dev/urandom of=disc_512/$i bs=1 count=$size
    dd if=/dev/urandom of=disc_1024/$i bs=1 count=$size
    dd if=/dev/urandom of=disc_2048/$i bs=1 count=$size
    dd if=/dev/urandom of=disc_4096/$i bs=1 count=$size

done
```