Zadanie 1.

- Ścieżka absolutna - ścieżka idąca od roota
- Ścieżka relatywna - ścieżka idąca od current working directory (katalog roboczy)
- Ścieżka znormalizowana (canonicalized pathname) - ścieżka z rozwiniętymi symbolicznymi dowiązaniami, odwołaniami do "/./", "/../" oraz mająca usunięte zbędne "/"

Każdy proces ma working directory od którego jest obliczana ścieżka relatywna. Możemy je zmienić syscallem

```
int chdir(const char *path);
int fchdir(int fd);
```

Montaż systemu plików oznacza udostępnienie tego systemu plików w pewnym punkcie drzewa katalogów. Ten punkt nazywa się **punktem montażu** (mounting point).

- **noatime** Do not update inode access times on this filesystem (e.g. for faster access on the news spool to speed up news servers).
- **noexec** Do not permit direct execution of any binaries on the mounted filesystem.
- **sync** The option sync means that all changes to the according filesystem are immediately flushed to disk.

Zadanie 2.

Działanie *rename* nie jest opisane w tym manualu, więc opiszę to swoimi słowami. Jak wiadomo, każdy istniejący plik ma swoje *dirent*, które na niego wskazują. Więc to co robi rename, to tworzy nowy *dirent* w miejscu docelowym, który wskazuje na tego samego *vnode*, a starego *dirent* usuwa. (Może ktoś potwierdzić?)

Atomowa operacja- operacja niepodzielna czasowo. Chociaż w tym kontekście chodzi pewnie o coś innego. Chodzi o to, że jeśli będzie *rename* na plik, który nie istnieje punkt w czasie, dla którego pewnien proces nie znajdzie pliku na miejscu docelowym. Co ciekawe, może się zdarzyć sytuacja, że podczas przenoszenia pliku zarówno stara pozycja jak i nowa będą wskazywały na ten sam plik. Zatem pod tym względem już rename nie jest atomowy.

"Czemu «rename» zakończy się błędem «EXDEV» kiedy próbujemy przenieść plik do

innego systemu plików?" Taka jest semantyka tego syscall'a. Dziwne pytanie. Dlaczego dokładnie to przypuszczam, że to jest spowodowane tym, że trzeba skopiować całego vnode, bo nie istnieją hardlinki pomiędzy róznymi systemami plików.

Zadanie 3.

- Dowiązanie symboliczne przechowuje ścieżkę do innego pliku. Ten plik może nie istnieć, ale symlink dalej będzie wisiał. Dereferencja dowiązania jest przezroczysta. Nie wykonujemy operacji na pliku dowiązania tylko na tym na co wskazuje
- Dowiązanie twarde to dodatkowy *dirent* na dany *vnode*.

```
ln -s a b
ln -s b a
```

Kiedy kernel sobie chodzi po ścieżkach pamięta on ile symlinków już przeszedł. Jeśli ta wartość jest za duża to zwróci ELOOP.

Dowiązań twardych nie można robić na katalogach, przez co nie istnieje możliwość pętli. Czemu nie można?

- system plików nie będzie już drzewem
- znormalizowane ścieżki nie będą jednoznaczne
- pliki by miały wielu rodziców
- liczba dowiązań katalogów::
  - +1 od mojego ojca
  - +1 ode mnie samego ("/./")
  - +1 od każdego z katalogu, który jest we mnie ("/../")