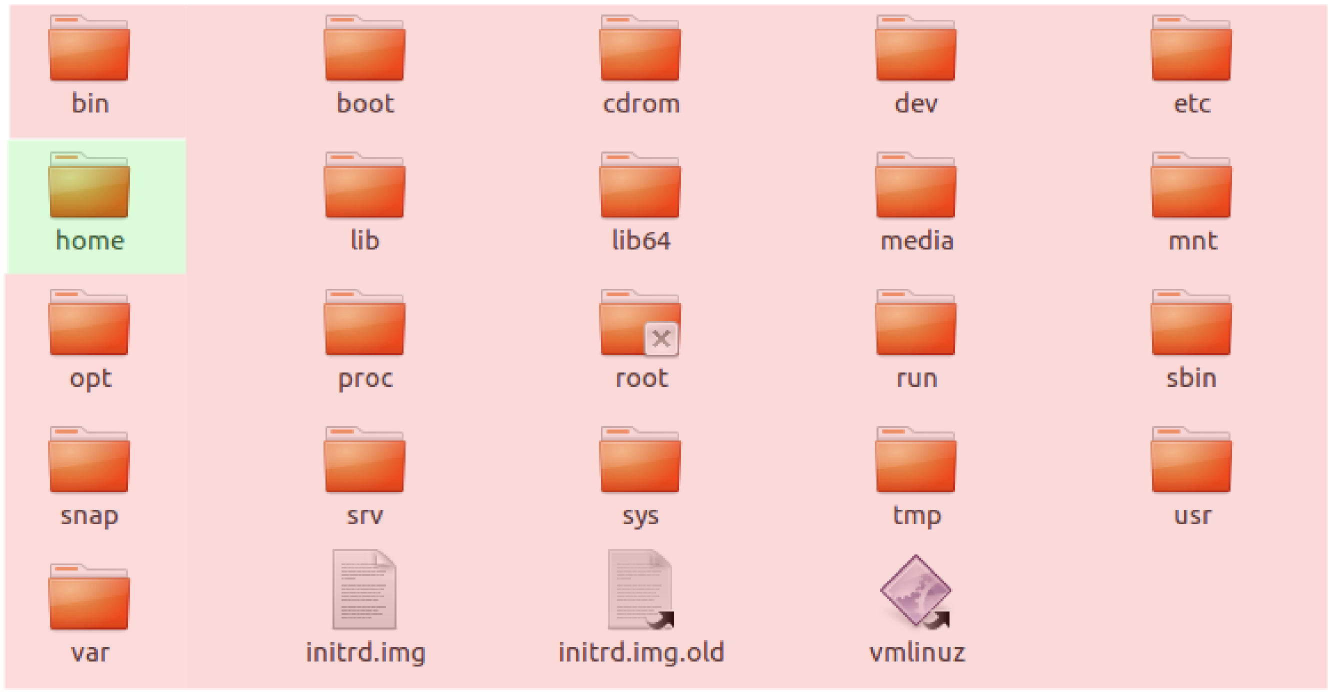
Linux Spaces

System-wise space vs. User space



- When working on your projects, you are a **USER**.
- When installing/upgrading system-wise application/library, you are an **ADMIN**.

Jumping between folders (changing directories)

```
$ cd (Relative Path|Absolute Path)
```

- In terminal commands, with `A|B`, I mean "Either A or B".

Listing files in the current directory (folder)

List files/directories inside the current directory of the terminal

```
$ ls
```

List files/directories on from other directory

```
$ ls (Relative Path|Relative Path)
```

Change folder name or moving folder name

```
$ mv (file|directory) (new file|new directory)
```

Copy file

```
$ cp (file) (target path)
```

Copy directory

```
$ cp -r (directory) (target path)
```

Create a new directory (folder)

```
$ mkdir (new folder name)
```

Removing a file

```
$ rm (file)
```

Remove a directory

```
$ rm -r (directory)
```

WARNING: Did you say `rm`?

HOW ABOUT `sudo rm -rf /`

DO NOT DO THIS!

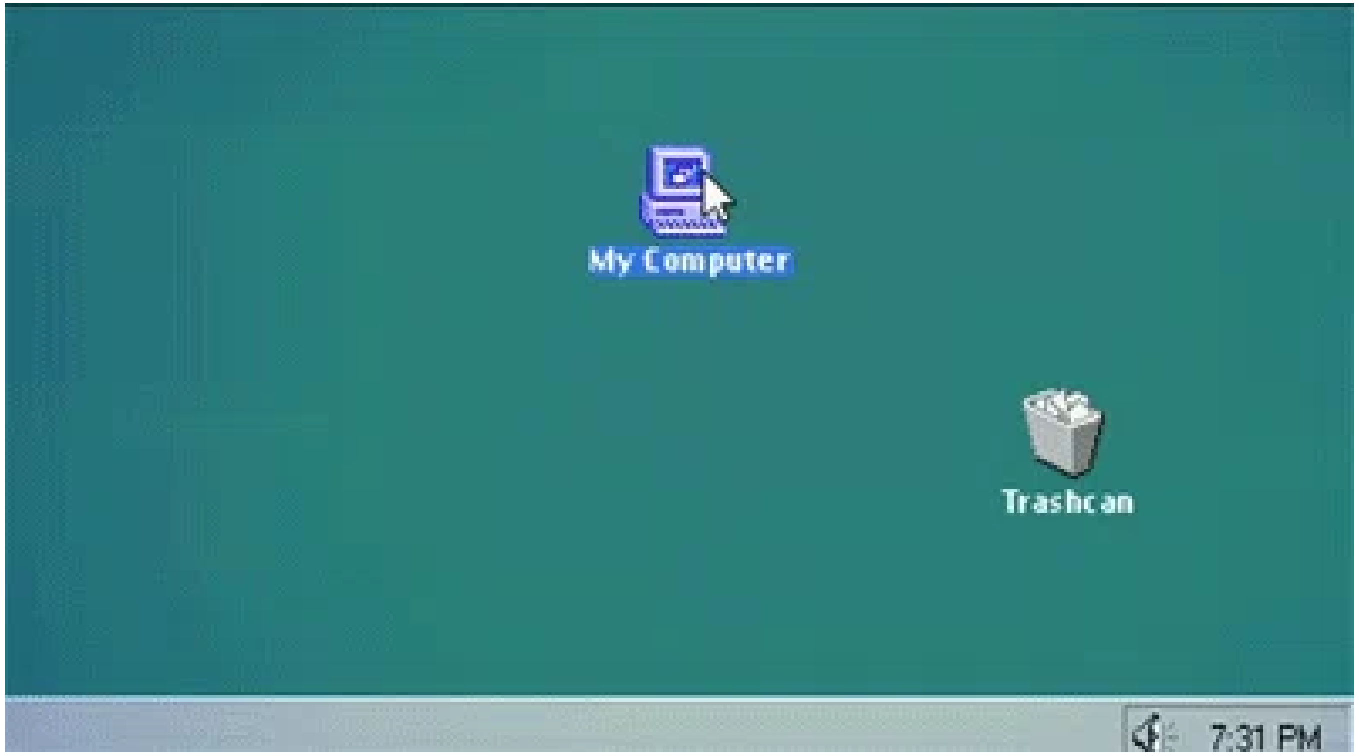
```
$ sudo rm -rf /
```

WARNING: Did you say `rm`?

HOW ABOUT `sudo rm -rf /`

DO NOT DO THIS!

```
$ sudo rm -rf /
```



Updating & Upgrading your Linux

Upgrades are very important. Many hardware drivers issues are being fixed through these updates. Also, security-wise, updates guarantees your system to be safe against hackable vulnerabilities. For example, *Spectre* and *Meltdown* vulnerabilities that exposed all Operating Systems (including Widnows and Linux), for more info.

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Installing packages from the apt store

```
$ sudo apt-get install (package name)
```

Installing local `.deb` packages

```
$ sudo dpkg -i (package path)
```

Interesting Appliactions

Category	**package name**
Music & Video	vlc, rhythm box (shipped with Ubuntu)
PDFs	Okular, Foxit, PdfShuffler
Screenshots	Shutter
C++ IDEs	Qt Creator, Jet-brains CLion, VSCode
Python IDEs	Pycharm, Anaconda (Spyder)
Web IDEs	VSCode, Jet-brains WebStorm

C++ Struct

Types in C++

C++ Struct

Types in C++

- Premitive Data Types (PDT), or first-class citizens, such as: `int`, `double`, `char`, etc.

C++ Struct

Types in C++

- Premitive Data Types (PDT), or first-class citizens, such as: `int`, `double`, `char`, etc.
- Custom, user-defined types, for example using: `struct` or `enum class`.

`struct` example

Consider the following application:

```
double area( double w , double h )
{
    return w * h;
}

int main()
{
    double w = 0, h = 0;

    std::cin >> w >> h;

    std::cout << area( w, h ) << std::endl;
    return 0;
}
```

`struct` example (cont'd)

Using `struct`:

struct example (cont'd)

Using struct:

```
struct Rectangle
{
    double w;
    double h;
};
```

`struct` example (cont'd)

Using `struct`:

```
struct Rectangle
{
    double w;
    double h;
};
```

- `Rectangle` is now a custom type,

`struct` example (cont'd)

Using `struct`:

```
struct Rectangle
{
    double w;
    double h;
};
```

- `Rectangle` is now a custom type,
- consists of two `double`s.

`struct` example (cont'd)

Using `struct`:

```
struct Rectangle
{
    double w;
    double h;
};
```

- `Rectangle` is now a custom type,
- consists of two `double`s.
- Think of it as a package.

struct example (cont'd)

struct example (cont'd)

```
struct Rectangle
{
    double w; // First member
    double h; // Second member
}; // Don't forget a semicolon here!
```

struct example (cont'd)

```
struct Rectangle
{
    double w; // First member
    double h; // Second member
}; // Don't forget a semicolon here!
```

```
double area( Rectangle rectangle )
{
    return rectangle.w * rectangle.h;
}
```

struct example (cont'd)

```cpp
struct Rectangle
{
    double w; // First member
    double h; // Second member
}; // Don't forget a semicolon here!
```

```cpp
double area( Rectangle rectangle )
{
    return rectangle.w * rectangle.h;
}
```

```cpp
int main()
{
    Rectangle rect;
    rect.w = 3;
    rect.h = 5;
    std::cout << area( rect ) << std::endl;
    return 0;
}
```

struct example (cont'd)

```
struct Rectangle
{
    double w; // First member
    double h; // Second member
}; // Don't forget a semicolon here!
```

```
double area( Rectangle rectangle )
{
    return rectangle.w * rectangle.h;
}
```

```
int main()
{
    Rectangle rect;
    std::cin >> rect.w >> rect.h;
    std::cout << area( rect ) << std::endl;
    return 0;
}
```