Can't Believe It's Linux

a totally different and hypothetical linux distribution

What's the Situation?

- ubuntu is doing pretty well
- OS X is doing a lot better
- crap is actually pretty cool
- nobody cares about freedom (!!)
- the web is kicking ass

What could happen?

- OS X becomes the next Windows
- OpenGL and DirectX will no longer matter
- The Web will become the next killer application
- Netbooks could be a small chance for Linux

What are the Problems?

- Too many desktop environments
- Too many libraries / frameworks
- Stupid community
- Apple / Microsoft have learned their lessons

The real Problems!

- dll hell – we've got it too, just worse
- the filesystem structure sucks ass
- no root? you're pretty much screwed
- the permissions are crap too!

John Doe's Problems

- After installing his system, John Doe downloads the flash player from the official website just to find out he has no idea how to compile software. There is no package he can install, no .dmg, no .exe

Say what you want, but without one established distribution system it just sucks for the desktop user.

John Doe's Problems

- John upgraded his system, and his four year old commercial music software is unable to run any more.

Turns out that shared libraries changed incompatible and the original developer would have to recompile it

You know that I can still run some of my old DOS games on Windows XP? (Haven't tried Vista)

John Doe's Problems

- John installed a software he downloaded from a webpage (that happened to be a software written in Qt) and it looks like crap on his system or totally different than the application's he used so far.

THANKS NOKIA, you're working on *that* problem... But it took a while and still isn't perfect

John Doe's Problems

- Joe is one of four users on the family's computer. He downloads a game from the network but isn't root so he is unable to use the system tools for application installation to install the application into the home directory.

No problem on the server, I know. But it sucks for normal users.

So what's the Plan?

Steal from OS X

(and Windows)

A new Filesystem Structure

/Home/<username> — Where users go

/System — Where the system goes

/ — Where global stuff goes

Why not lowercase? We're on a desktop, users might look at it. And "/System/Configuration/Grub/menu.lst" looks a lot nicer than "/sys/config/grub/menu.lst".

What's below that

Below each of those three folders we have the same directory structure:

./Library	Shared Libraries of all Languages
./Applications	Symlinks to executable Binaries
./Extensions	Plugins for applications
./Configuration	Application Configuration
./Volumes	Mounted Media

Where's /dev?

The following folders can stay where they are:

/tmp
/dev
/boot
/root

no need to move them around. But GUI applications like nautilus should probably hide them by default.

A Package

... is some sort of archive with all the files in it. We have pretty cool package systems already, take advantage of them and extend them so that they can install into different locations:

/Home/<username>
/System
/

The User Interface

aka: The ugly part. I know there are so many GNOME haters, but there are many reasons why GNOME is the best pick:

- Companies stand behind it
- huge momentum currently
- The core libraries are all written in C
- The user interface is very user friendly

Are you crazy?

No. And here's why:

For a long time Linux distributions shipped a shitload of text editors, file managers, cd burning applications etc. Many of them where part of the out-of-the-box installation and users [and by users I mean me] was impressed but confused.

Too many different approaches, no experience what application to use in what situation.

Then ubuntu came along and shipped just one of each and it become a success.

Make your Choice

- The user just wants his work done, and if it's called KDE or if it's called GNOME: it doesn't matter.
- As long as the applications have a similar appearance, the same shortcuts and if there is just one of each kind.
- Pro-users will always install more and different stuff if the want.

What's below /Libraries?

./Firefox	The application / shared lib
./Firefox/3.0	One specific version
./Firefox/Current	The version the user selected

If there are globally Firefox 3.0 and Firefox 2.0 and the "Current" version globally is 3.0, the user can symlink the global 2.0 folder as ~/Library/Firefox/Current and he will use 2.0

Where's the binary?

./Applications/firefox
-> ./Library/Firefox/3.0/Current/bin/firefox

All the applications a user might want to use are symlinked into his personal ./Applications folder. The global symlinks from /Applications are combined into that folder with some magic. Similar to how Live CDs overlay the root file system with a memory filesystem.

Where does it look?

Applications (binaries of applications):

~/Applications/<name>

Libraries:

~/Library/<name>/<version>
/Library/<name>/<version>
/System/Library/<name>/<version>

Configs?

A bit more tricky. What are configuration files about?
One the one hand you want the defaults globally available, on the other hand you want to overlay changes individually:

~/Configuration/<app>/<config>
/Configuration/<app>/<config> **[or]**
/System/Configuration/<app>/<config>

On Configs

Applications have to read both the global config and override the values with the values from the local config on a per-key/value basis.

And I don't think gconf is the solution.

The FHS

... is user unfriendly

... can be changed if the applications adapt. See OS X

... doesn't give you much

INTERMISSION

The Shell

The Linux shells suck. I'm sorry. Every time I write a quick shell hackery I know it will break if the filenames are too long, if there is unexpected whitespace somewhere, I can't properly escape parameters which gives me security nightmares etc.

The Modern Shell

Imagine a shell like this:

```
$ ps | filter { $_.mem -gt 500MB } | kill
2 processes killed
[<Process firefox-bin>, ...]$o1
$ foreach $o1 { echo $_.name }
firefox-bin
ooo-bin
```

- It's passing real objects around
- Commands (`ps`, `kill`) are classes in Mono assemblies
- Commands return new objects that are flagged with unique identifiers and kept in memory for some time (`$o1`)

Implementation Problems

- nobody will stand behind such a project
- it will be incompatible with everything out there
- the open source community will hate it
- it will take a long time to get all the builtin applications work with that dramatically different structure

</daydream>

now think about it ;)