Virtualised filenames – an alternative to docker.

Portable vs non-portable applications

Portable application

```
If (Posix API avaialable or Win API avaialable) {
    Take user input;
    Process user input;
    Produce output;
}
```

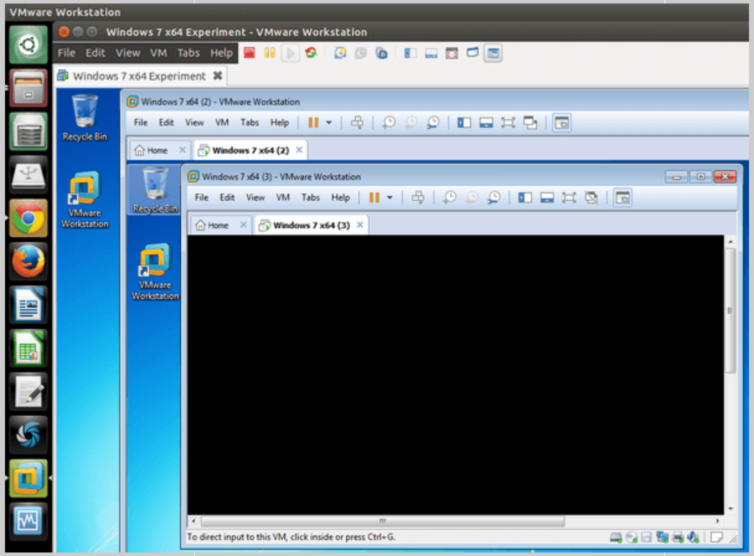
Non-portable application

```
If (not on x86) goto fail;
If (not on linux) goto fail;
If (not on linux 3.4+) goto fail;
If (libc is not glibc) goto fail;
If (glibc is not version 2.21+) goto fail;
If (/usr/bin/python not there) goto fail;
If (/usr/bin/python --version is not exactly 2.7.10) goto fail;
If (current executable's filepath is not in /usr/bin) goto fail;
If (you're lucky enough that none of the above)
{
    Take user input;
    Process user input;
    Produce output;
}
```

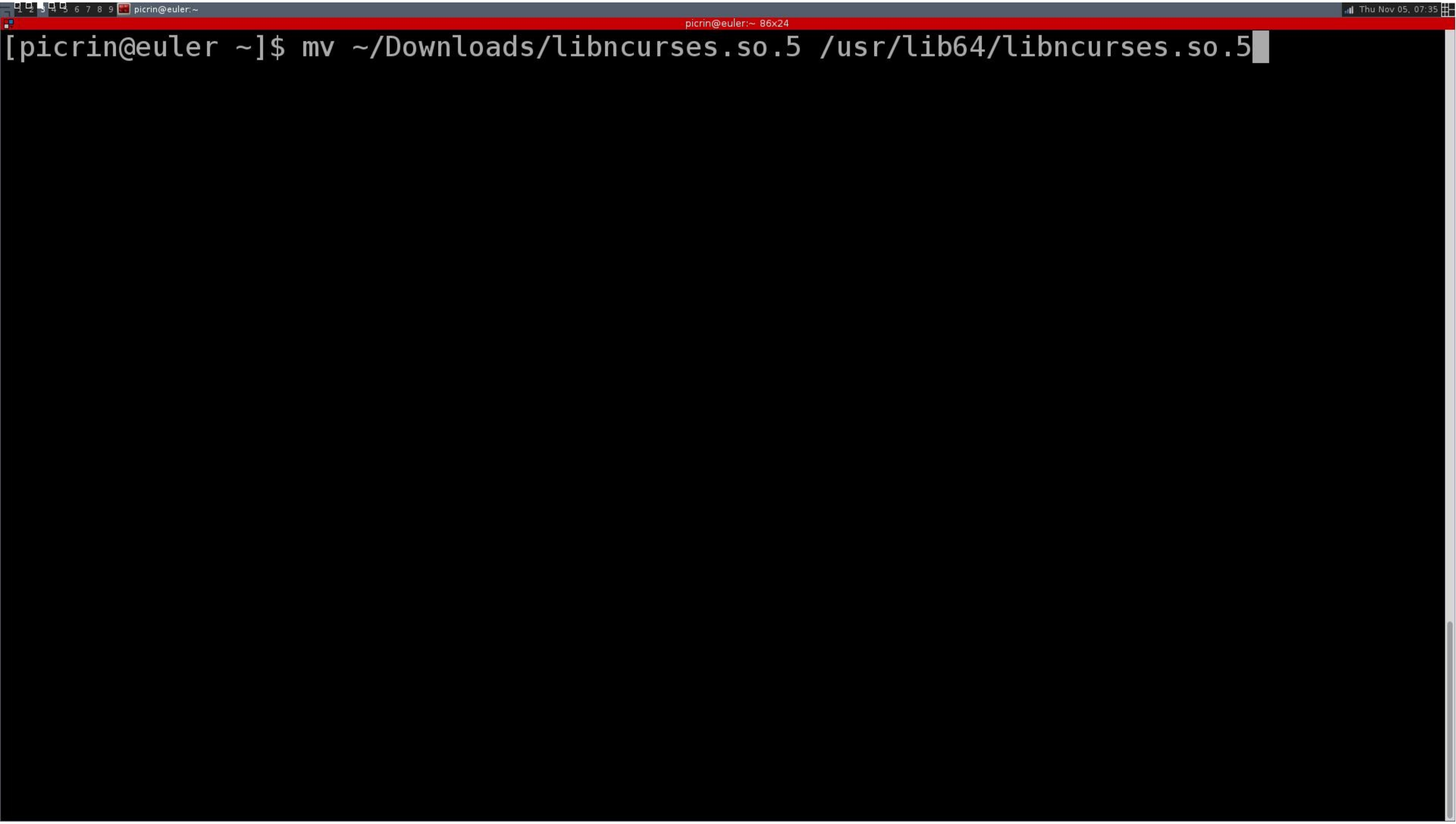
I want this program installed & running by Monday

most unportable < least unportable

Buy new (possibly old) hardware	Emulate the architecture	Emulate the OS	Emulate fs, network stack and process tree.

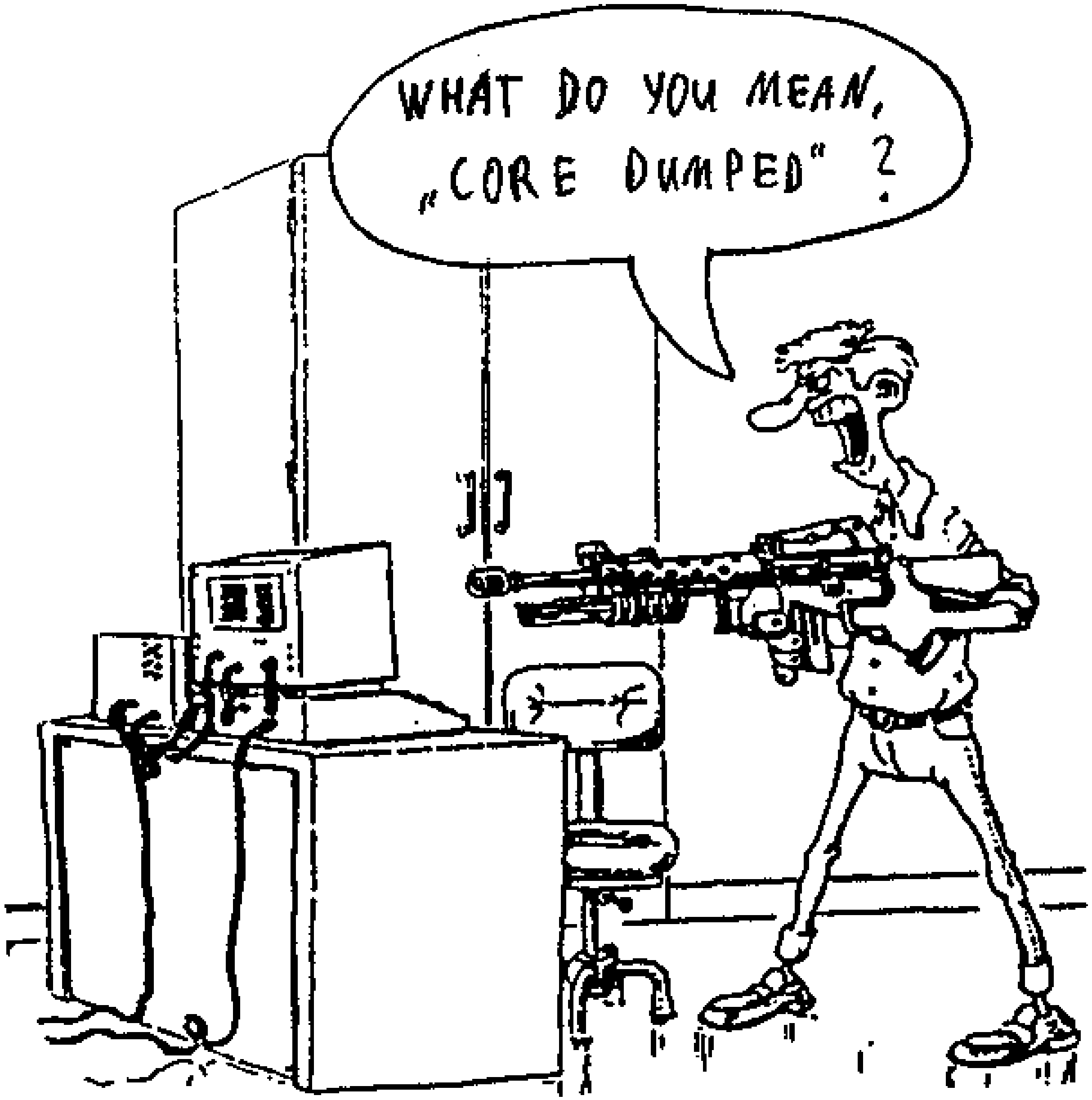

VMware Workstation
Windows 7 x64 Experiment - VMware Workstation
File Edit View VM Tabs Help
Windows 7 x64 Experiment
Windows 7 x64 (2) - VMware Workstation
Recycle Bin
VMware Workstation
Home
Windows 7 x64 (2)
Windows 7 x64 (3) - VMware Workstation
Windows 7 x64 (3)
To direct input to this VM, click inside or press Ctrl+G.

There is one more way...

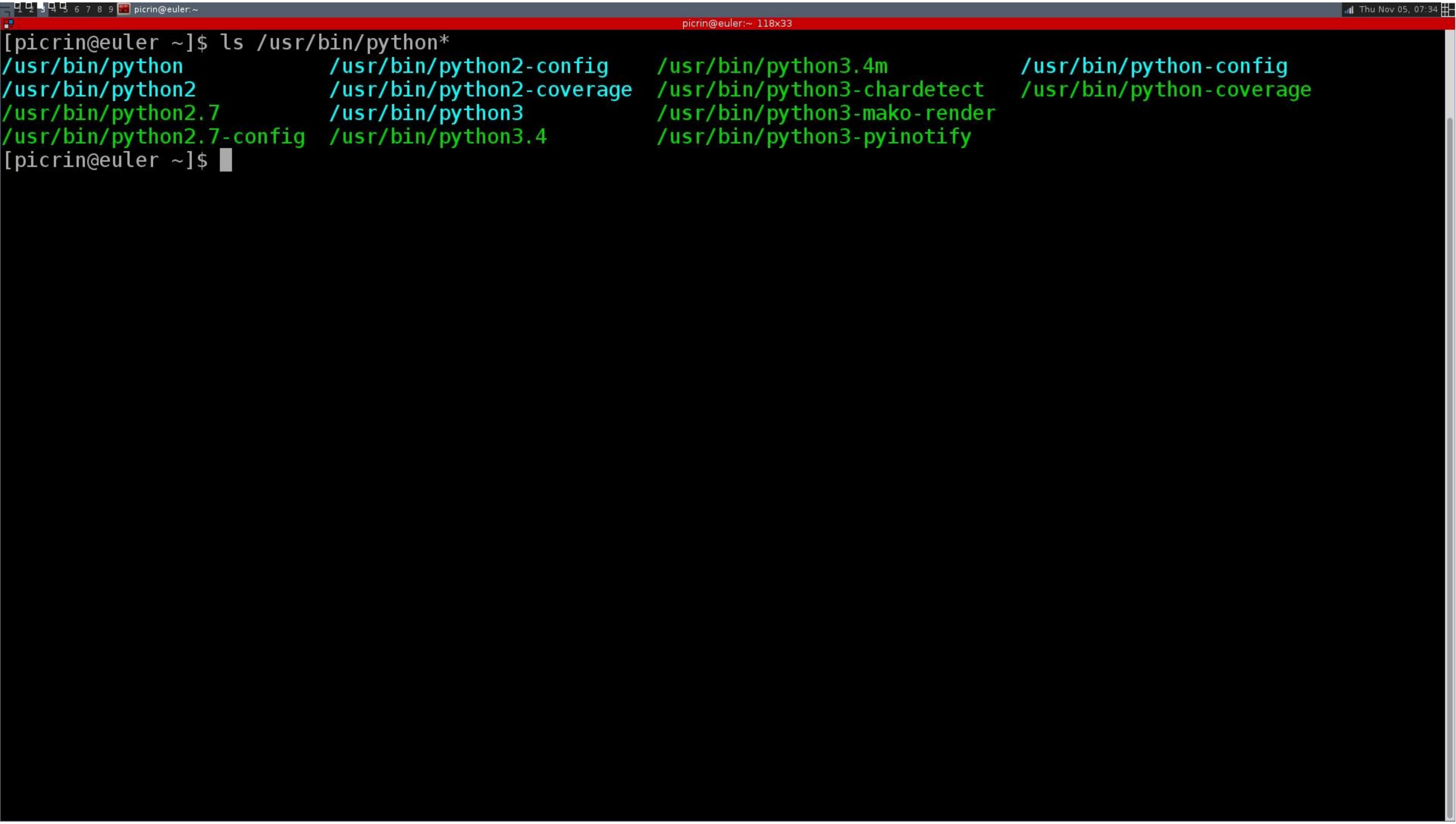


```
[picrin@euler ~]$ mv ~/Downloads/libncurses.so.5 /usr/lib64/libncurses.so.5
```

Painful way

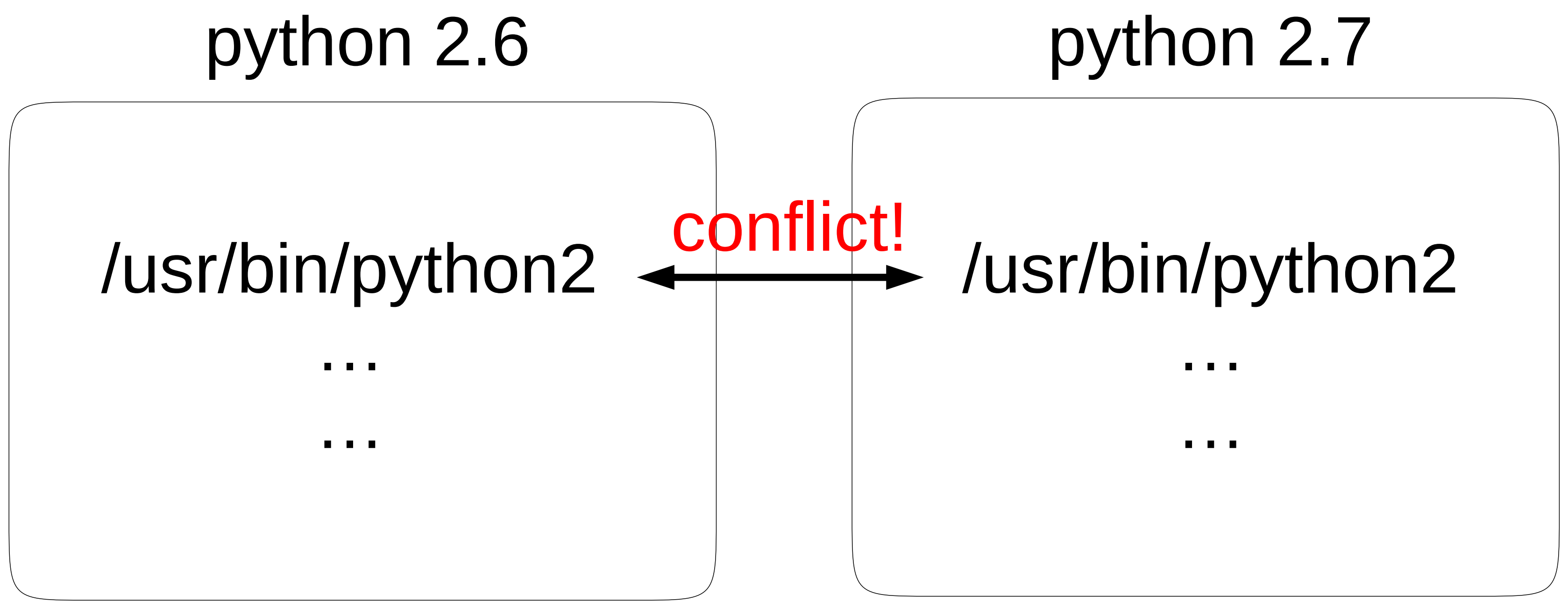


Rpm/dpkg to the rescue?



```
[picrin@euler ~]$ ls /usr/bin/python*
/usr/bin/python             /usr/bin/python2-config    /usr/bin/python3.4m            /usr/bin/python-config
/usr/bin/python2            /usr/bin/python2-coverage  /usr/bin/python3-chardetect    /usr/bin/python-coverage
/usr/bin/python2.7          /usr/bin/python3           /usr/bin/python3-mako-render
/usr/bin/python2.7-config   /usr/bin/python3.4         /usr/bin/python3-pyinotify
[picrin@euler ~]$
```

Not Really


python 2.6
/usr/bin/python2
…
…
conflict!
python 2.7
/usr/bin/python2
…
…

Hack into libc a new call vfn

A process makes a call vfn(“/usr/bin/python”, “/home/picrin/python2_5”)	A process doesn't make the vfn call
From now on each call to open(), stat(), exec(), etc. will execute as normal, EXCEPT if path is /usr/bin/python, in which case they'll execute as if path was /home/picrin/python2_5	each call to open(), stat(), exec(), etc. will execute as normal.
Each process can execute vfn multiple times, to virtualise as much (or as little) of the filesystem as it wishes. Each process inherits the vfn translation table from its parent process.	No process can alter vfn translations of any other process (security). You don't have to be root to alter your own process's vfn translation table (sensible). Nitty-gritty details-pitfalls, like executing vfn on files with open descriptors, etc. vfn translation table possibly has to be implemented inside kernel

vfn to the rescue!

```
rpm -ql python3
```

```
/usr/bin/pydoc3
/usr/bin/pydoc3.4
/usr/bin/python3
/usr/bin/python3.4
/usr/bin/python3.4m
/usr/bin/pyvenv
/usr/bin/pyvenv-3.4
```

```
rpm -ql python3 | xargs sha256sum
```

```
 /usr/bin/pydoc3 -> /opt/vfn/8ff8790221fbf907c4c6e2db127664ec59f47e9bfb659252441900dbcc0dccaf
/usr/bin/pydoc3.4 -> /opt/vfn/8ff8790221fbf907c4c6e2db127664ec59f47e9bfb659252441900dbcc0dccaf
/usr/bin/python3 -> /opt/vfn/4b78808d15be5bebfa7e0d11410d3bca84e370a196ec06d144ed214540b4060f
/usr/bin/python3.4 -> /opt/vfn/4b78808d15be5bebfa7e0d11410d3bca84e370a196ec06d144ed214540b4060f
/usr/bin/python3.4m -> /opt/vfn/4b78808d15be5bebfa7e0d11410d3bca84e370a196ec06d144ed214540b406
/usr/bin/pyvenv -> /opt/vfn/9e64cc7d0101933c7288e1866f5a8d70ddef8a121308a4faa987c4c0c00a254c
/usr/bin/pyvenv-3.4 -> /opt/vfn/9e64cc7d0101933c7288e1866f5a8d70ddef8a121308a4faa987c4c0c00a254c
```

yeah!

Packaging becomes easy.

Dependency reuse becomes easy.

Software shipping becomes easy.

Containers become secure (no UID=0).

You no longer need UID=0 to install anything anyware.

Writing software is still hard ← focus there, portability is still a big bonus!

wait a minute

Need own kernel module to keep the vfn translation table, work out table inheritance, etc.

Need to hack EVERY libc call, which does a filesystem operation.

Some software STILL compiles against kernel headers, no way to do translation there.

There are multiple libc's to hack (glibc, bionic, bsd, etc.).

What about windows?