GPL and Source Code

Tatsuo Nakajima

Waseda University

tatsuo@dcl.info.waseda.ac.jp

Contents

- GPL and LGPL
- Program: Technical aspects
- Program: License aspects
- Embedded system and Linux
- Japan Embedded Linux Consortium

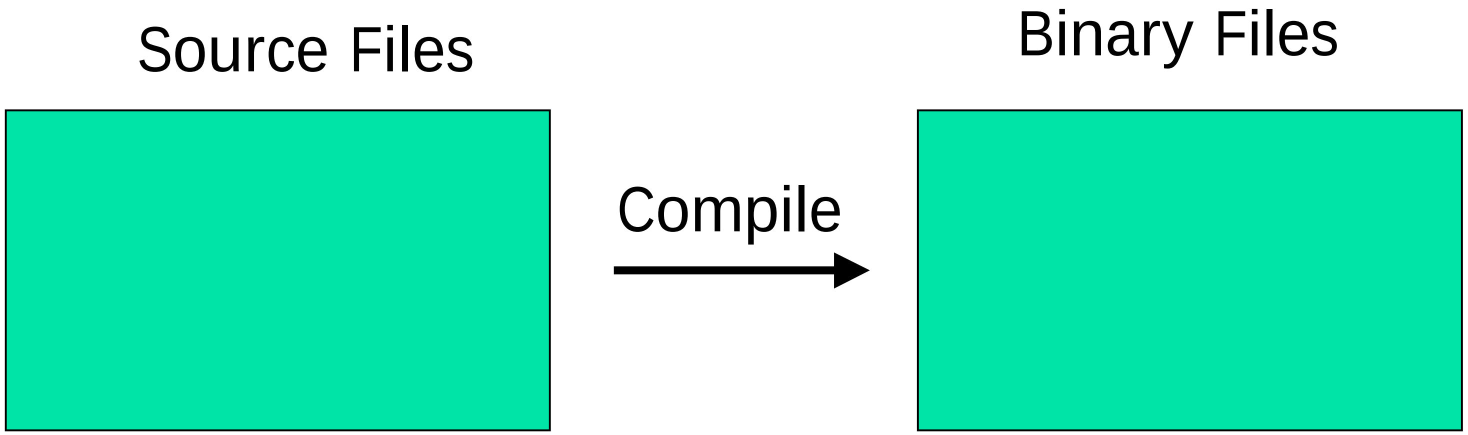
Social Issues and Computer Science

- We need to take into account various social issues in computer science.
  - License
    - Programs usually cannot be used freely.
  - Trust
    - Who I believe?
  - Privacy
    - How to protect information about myself?

GPL and LGPL

- GPL
  - General Public License
  - GPL is usually used for applications, kernels and servers.
- LGPL
  - Lesser General Public License
  - LGPL is usually used for libraries.

Program(1)



Program(2)

Program

```
#include "header.h"

main(int argc, char **argv)
 {
   libcall();
 }
```

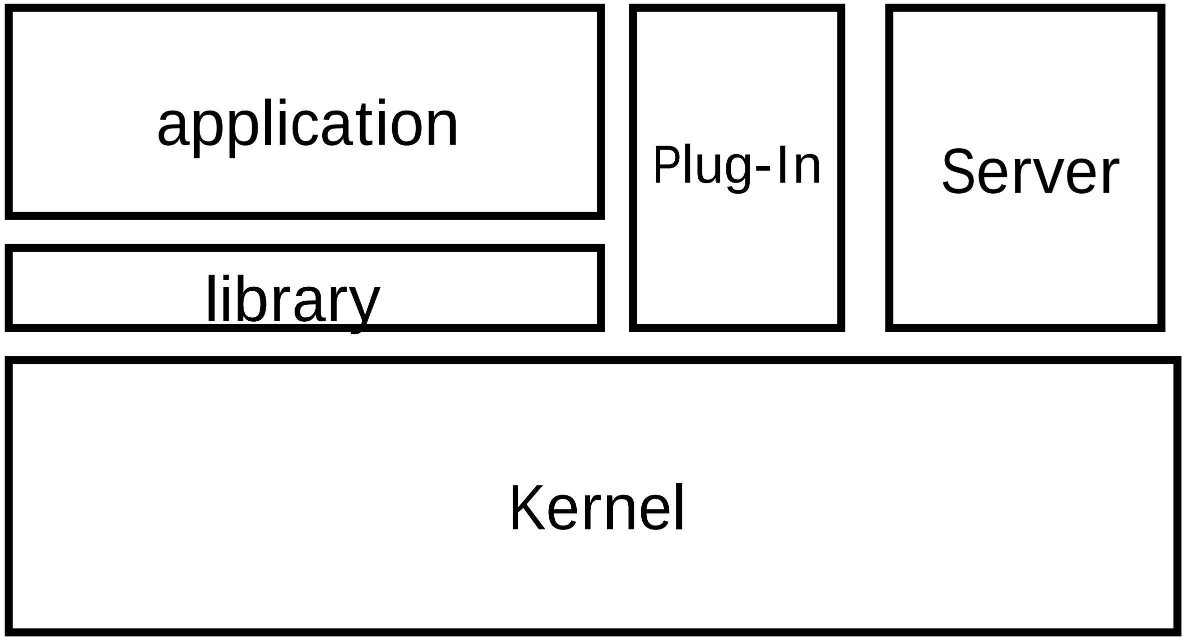
Library

```
void libcall()
{
  …
}
```

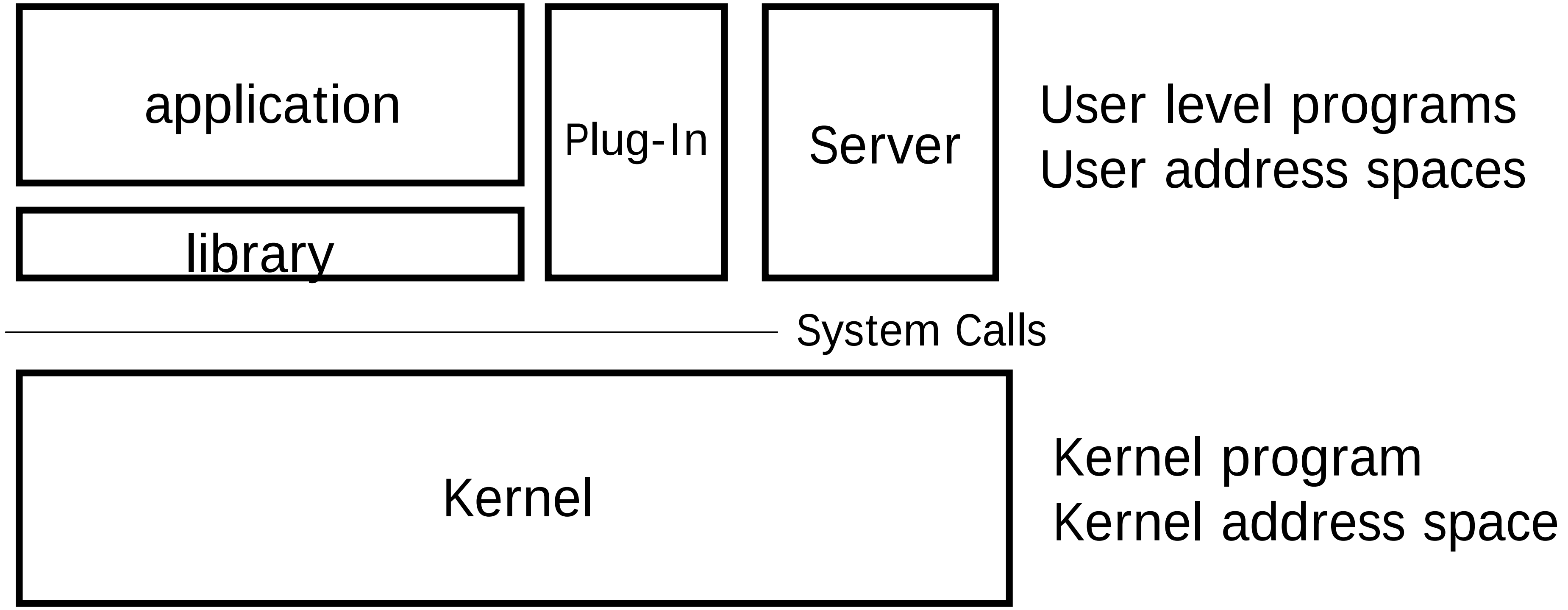
Header

```
#define MAX 10
#define max(a,b) ((a>b)?a:b)
```

Linux Structure



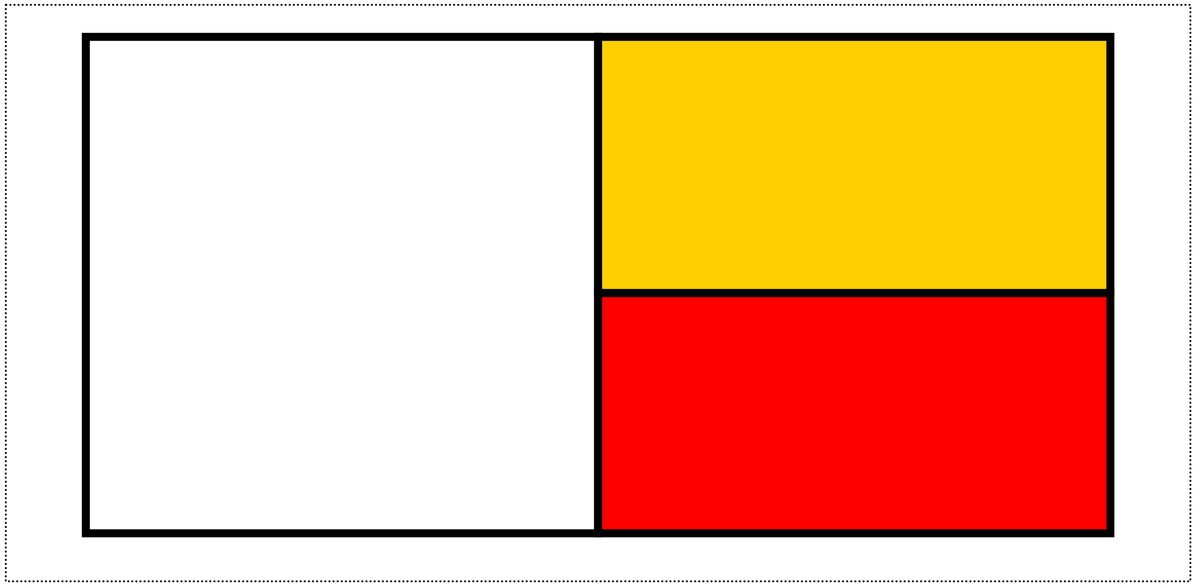
System Call



- ★ System calls in interface between user level programs and kernel.
- ★ System calls uses the trap instruction to switch address spaces.

Library

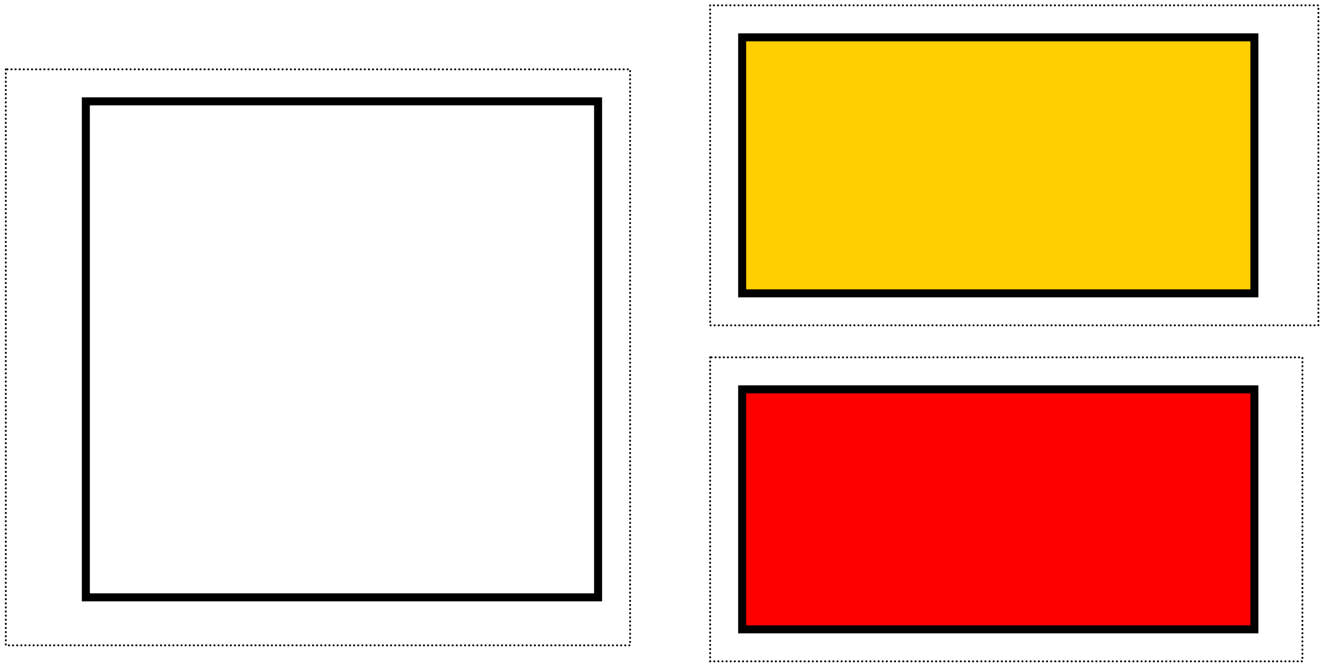
- Application programs uses many library programs to build executable codes.
  - glibc, GTK++, Qt,
  - cc program.c –o program –llib (liblib.a)



In one address space

Plug-In

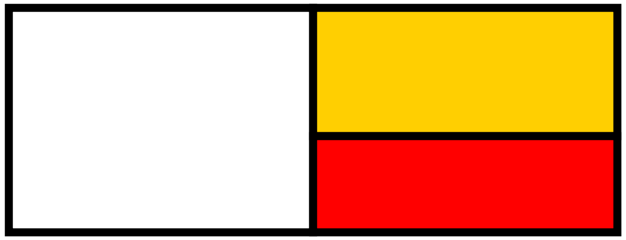
- Plug-ins are executed in different address spaces.



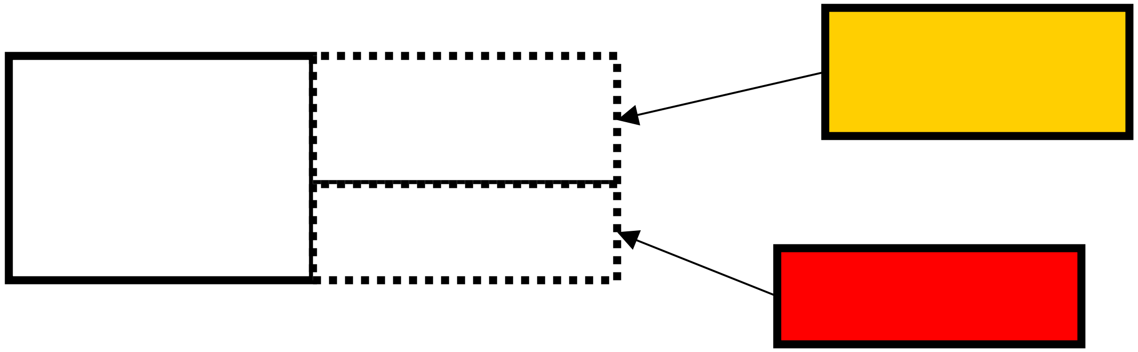
Many applications and servers

Linking

- Static linking
  - Libraries are linked at compile time.

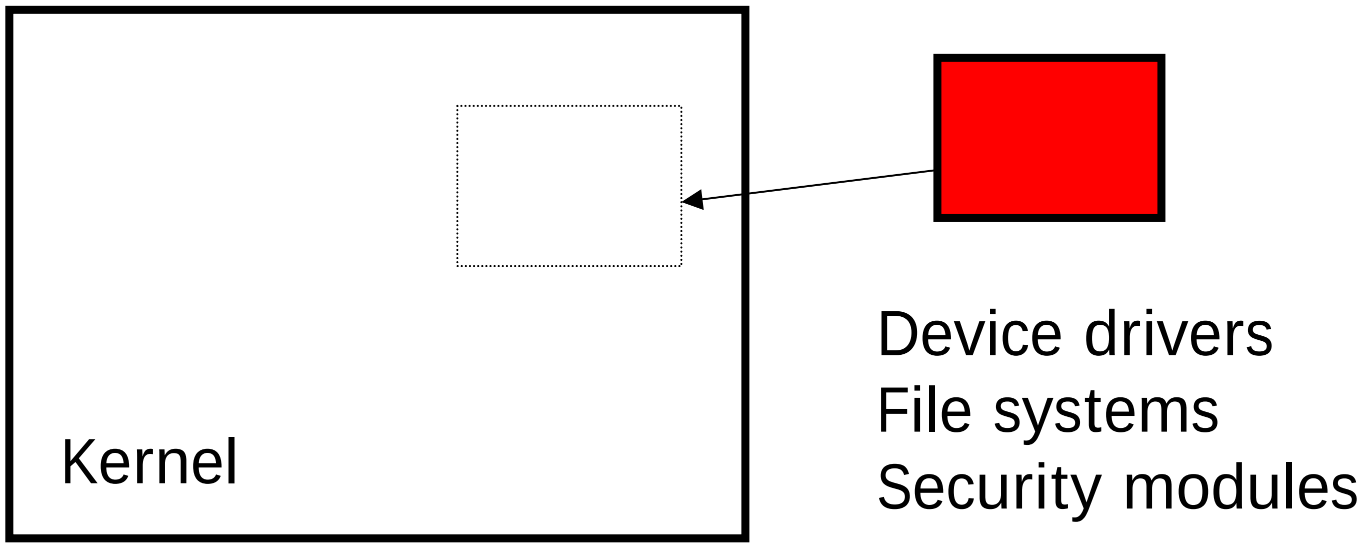


- Dynamic linking
  - Libraries are linked at run time.



Loadable Kernel Modules

- Loadable kernel modules can be linked in kernel at run time.
  - The strategy is similar to dynamic linking.



License Issues

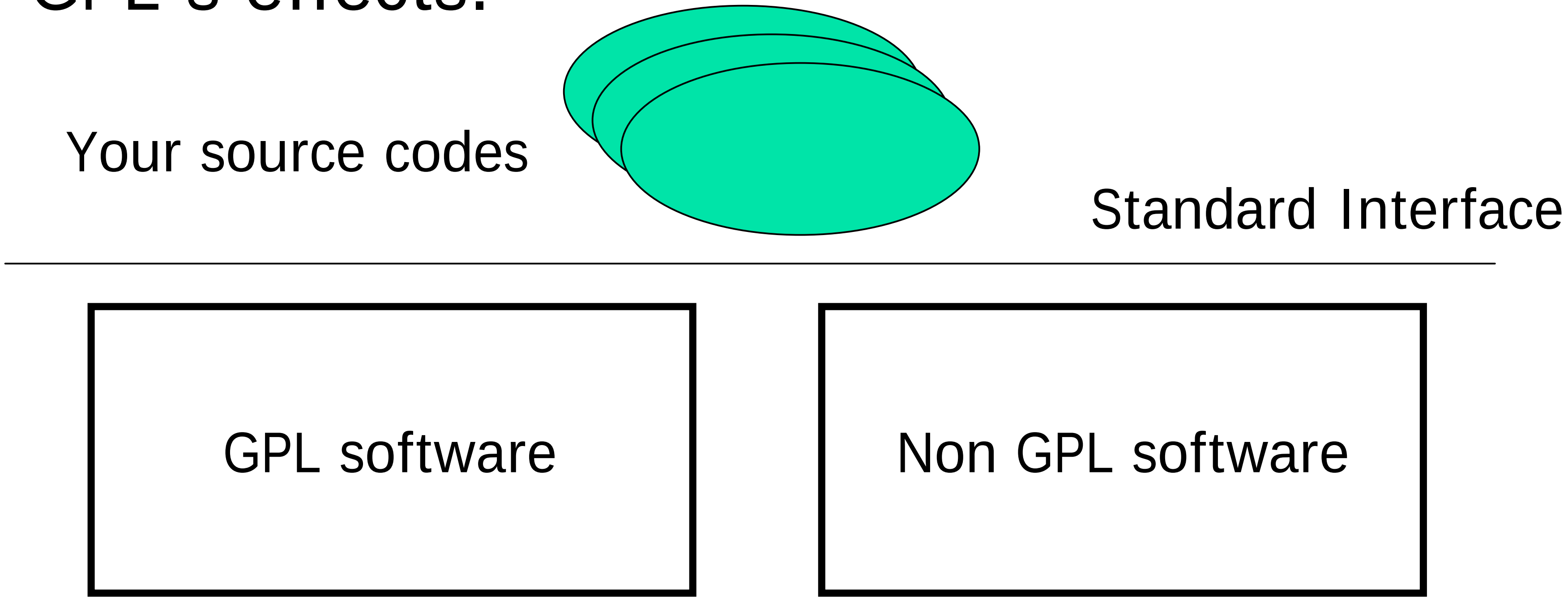
- GPL and Source codes
  - If you like to distribute your binary codes, you need to give their source codes if someone requests.

  → If you have no plan to give your binary codes to someone, you do not need to open your modified source codes.

  → Embedded systems contains binary codes, so, their source codes should be prepared to make them open.

Standard Interface

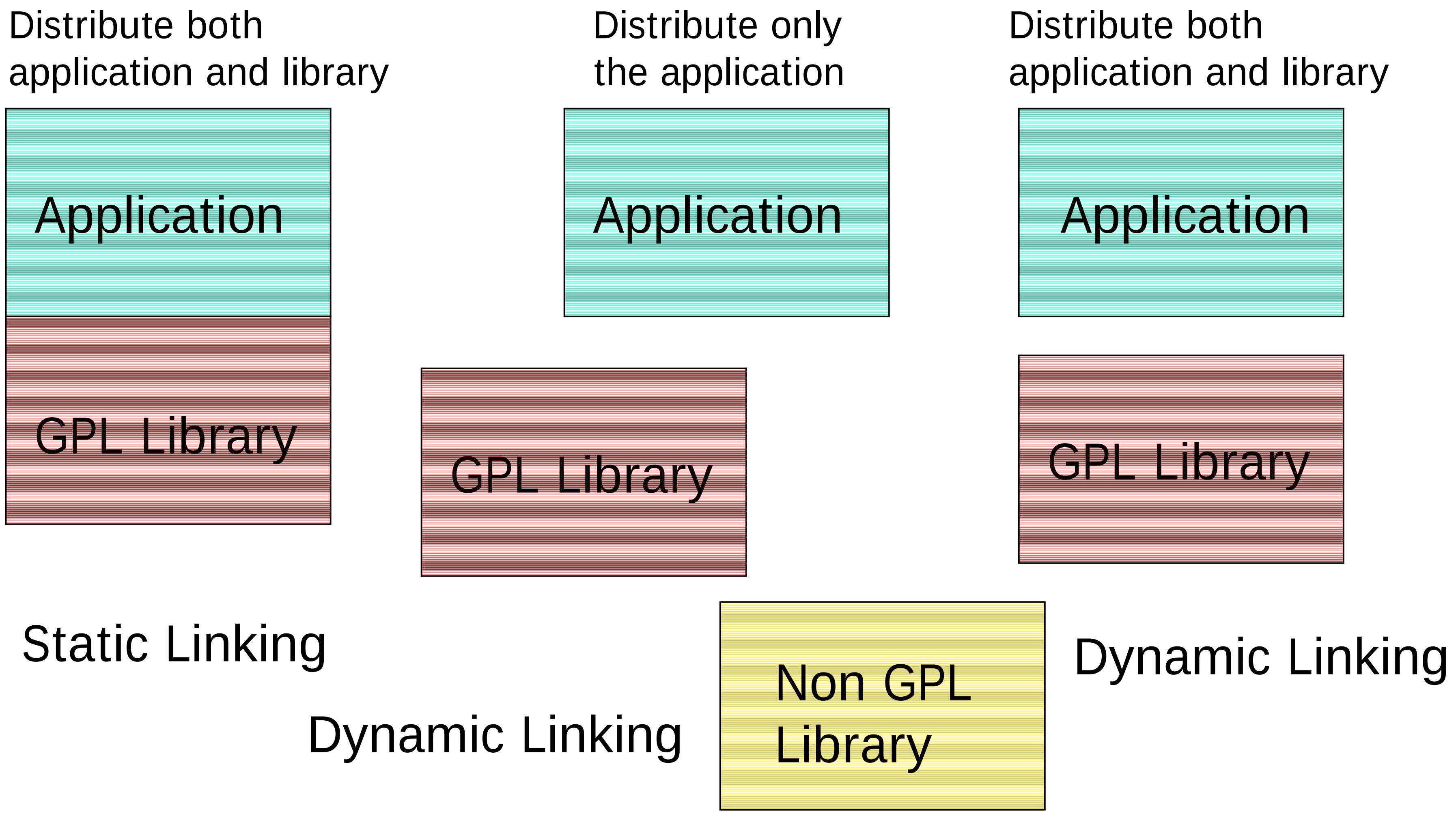
- Standard interface is a boundary to stop GPL’s effects.



Applications

- If Linux kernel interface is a standard interface, all applications can be proprietary.
    - The current Linux interface is based on POSIX, but not standardized.

Derivative and Linking(1)

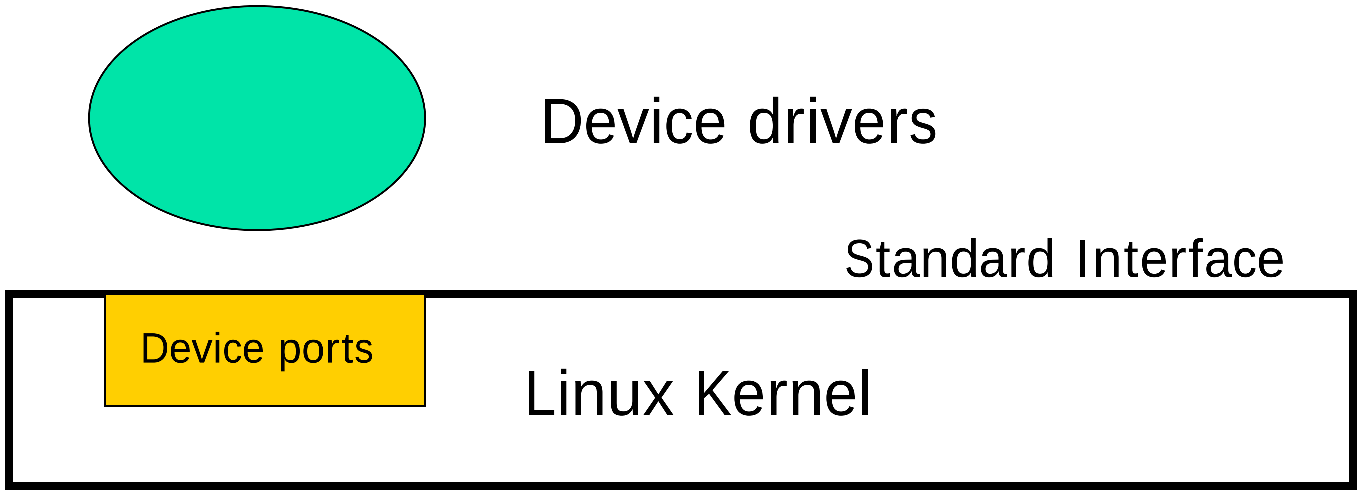


Derivative and Linking(2)

- If libraries offer standard interface, your source codes can be proprietary.
  - However, your binary code links GPL libraries statically, your code should be licensed by the GPL license.
  - But, you use dynamic linking, your code can be proprietary because the libraries can be replace with non GPL libraries.
  - But, if you distribute your code with GPL libraries in one system(ex. Embedded systems), your code may be licensed by the GPL license.
  - If a library is licensed by the LGPL license, your code can be proprietary.

User Level Device Drivers

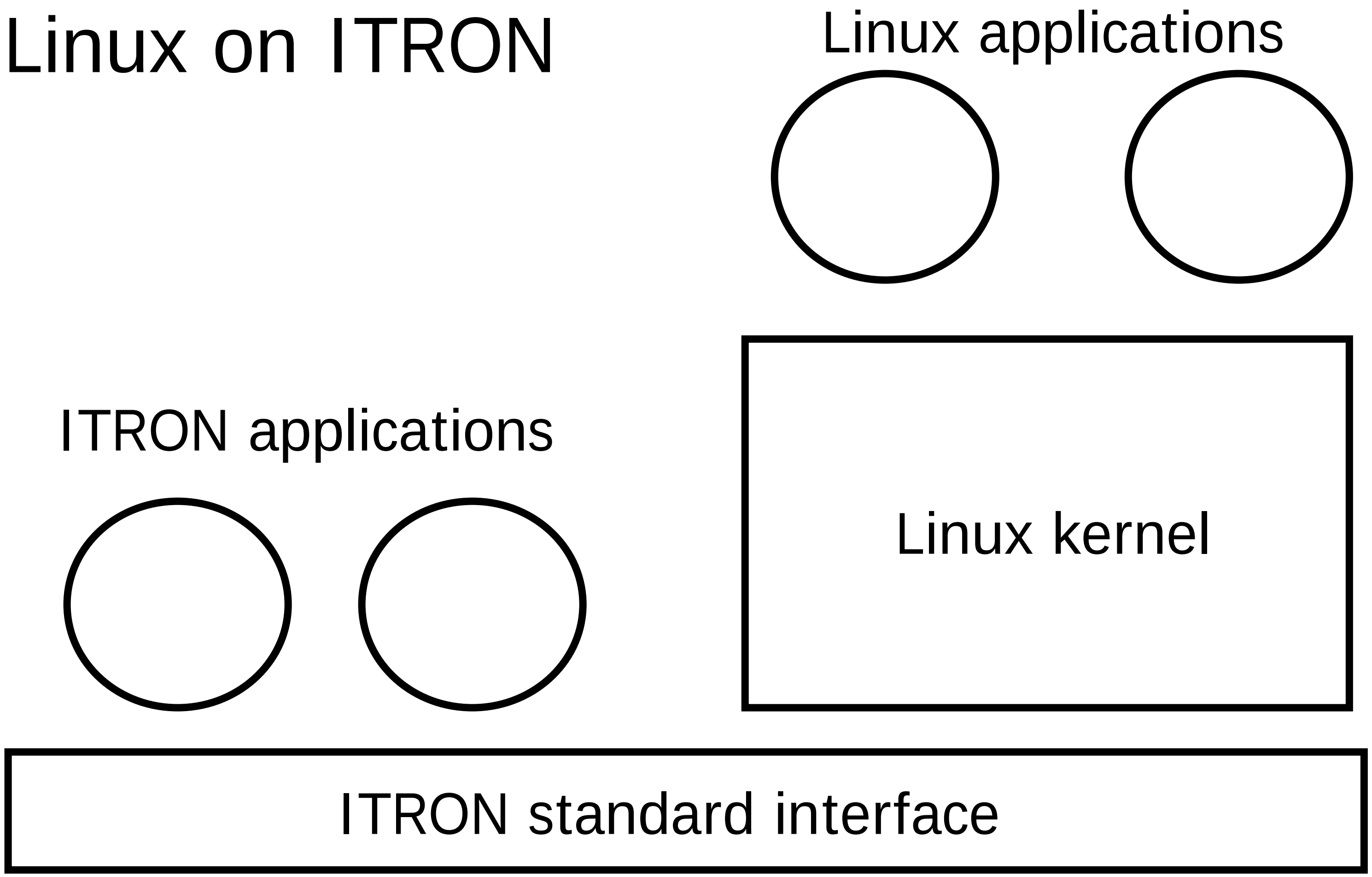
- User level device drivers are executed at the user level, and can be licensed by the GPL license.



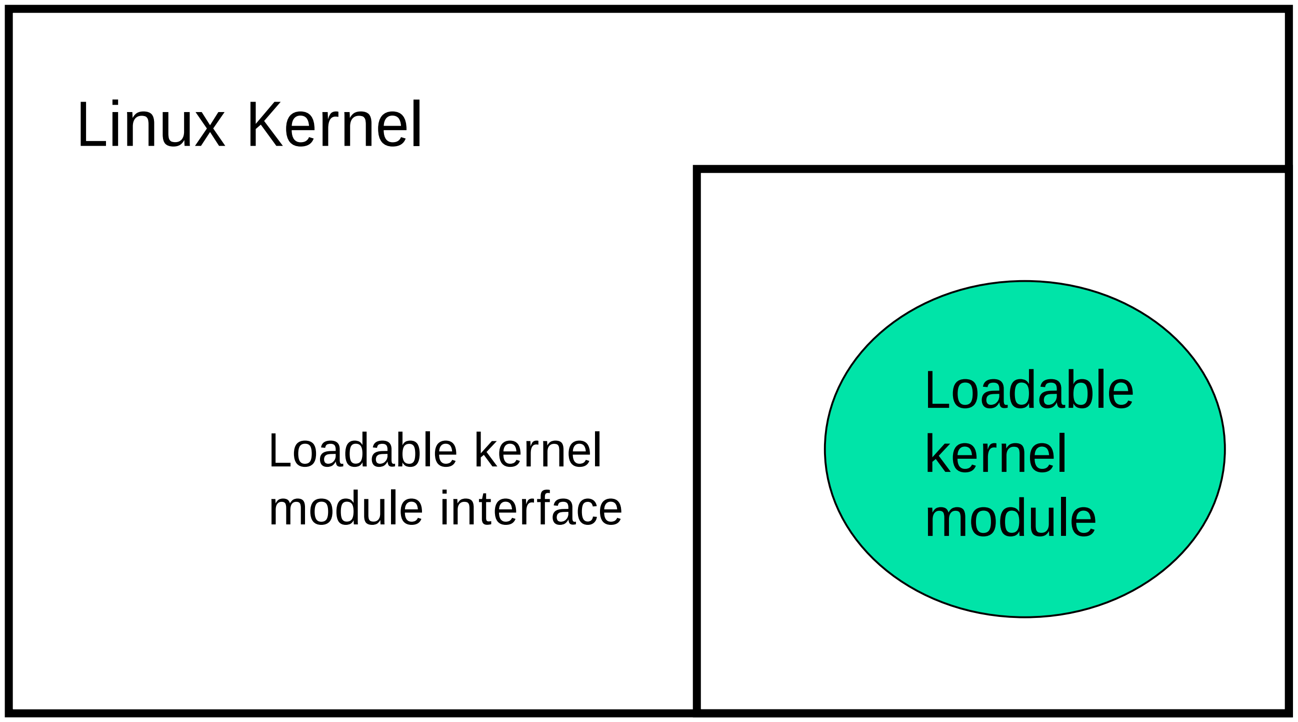
How to handle interrupts ?

Hybrid Architecture

- Linux on ITRON

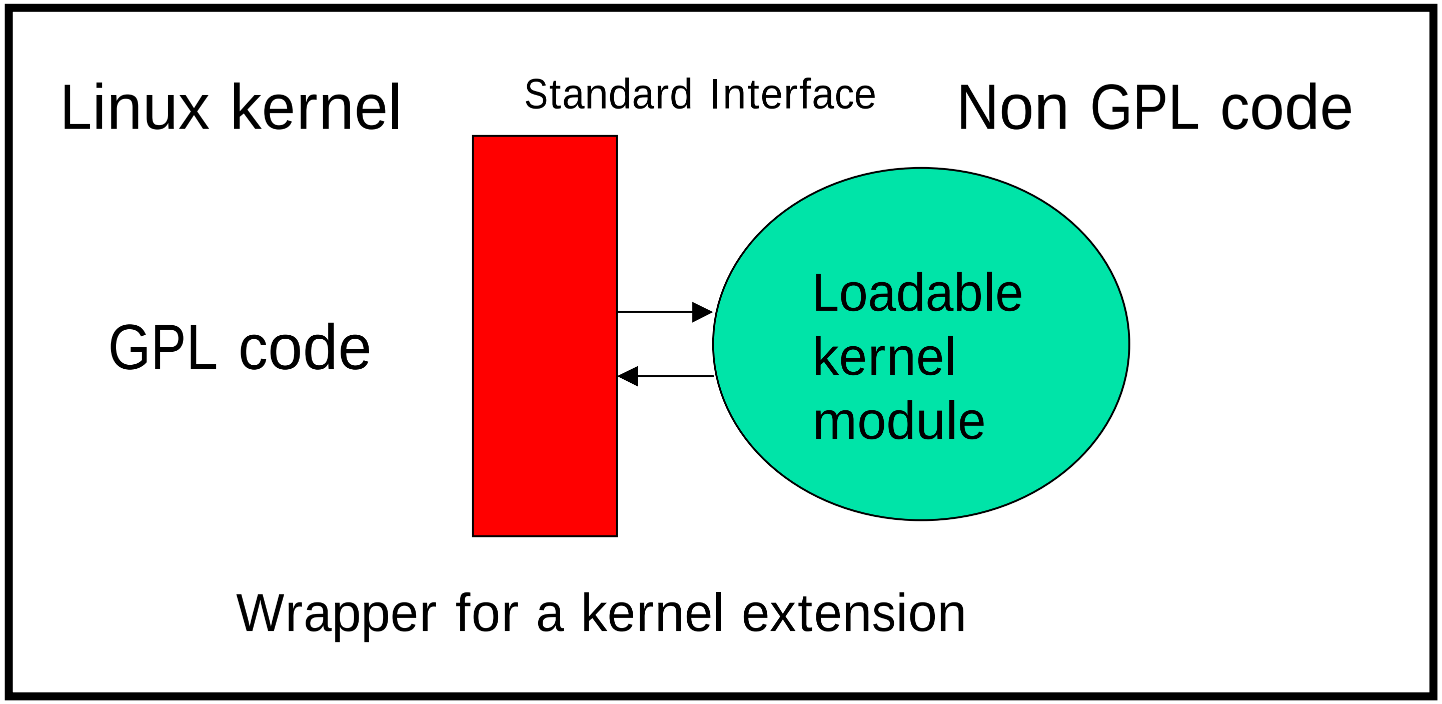


Loadable Kernel Modules



Loadable kernel module interface is not standard interface

Kernel Extension



Is this type of kernel extension possible ?

Discussions

- The definition of standard interface is not clear.
  - Is de-facto standard real standard ?
- The interpretation about GPL may be different in different countries.
- The discussions may be changed according to adopted operating systems.
  - Relying on kernel's extension mechanisms.
- The discussions may be changed according to adopted programming languages.
  - Relying on languages' functionalities.

The Current Status of Embedded Systems

- Embedded systems become more and more complex.
  - Mobile phones, Car navigation, Digital TV
- What problems?
  - The complexity is the same level of personal computers.
  - Traditional software infrastructures are not suitable to develop complex software.
  - We have new requirements every day.

Future Embedded Systems

- Deeply embedded
  - Simple and distributed
  - Severe resource constraints
  - Strict reliability, Real-Time
- Information Appliances
  - Specialized functions.
  - Complex, composable
  - Security, virtual reliability
  - Will replace current personal computers

Information Appliances

- I likes to use a display of a kiosk terminal from my PDA.
- I like to use a keyboard to edit data on my PDA.

→ Future computers are used to compose various devices.

→ Information appliances need protocol stacks, service discovery, Web services, dynamic program loading…

→ Memory protection, network support, POSIX support, various middleware.

→ Linux is the most suitable platform for information appliances.

Japan Embedded Linux Consortium(1)

- A consortium for supporting academic/industrial collaboration about embedded Linux
  - About 100 members
    - Matsushita, Sony, NEC, Fujitsu, Toshiba, Nokia Japan….
  - Promoting embedded Linux
  - Standardization of Embedded Linux related technologies.
  - Exchanging information with various consortiums.
    - Embedded Linux Consortium, CE Linux forum…
  - http://www.emblix.org/

Japan Embedded Linux Consortium(2)

- Open Source Issues
- License Issues
- Technical Issues
- Educational Issues

Open Source Issues

- Who maintains source codes ?
  - Budgets, License, sustainable services…
- Who writes programs ?
  - Budget, License, education…
- How to modify source codes ?
  - Community or Maintaining patches.
- How to use programs ?
  - Educational issues(books, tutorial)
- How to ensure the correctness of source codes ?
  - Testing.

License Issues

- License issues are more complex in embedded systems.
  - Libraries are distributed with proprietary application codes.
  - Dynamic loadable modules should by shipped with products.
  - It is difficult to ask users to install software.

Conclusion

- The presentation describes technical issues related to GPL.
  - GPL does not define technical issues.
    - Linking strategies, program structure…
  - We need to interpret GPL in respective situations.
  - Currently, each person may have different interpretation.