B4B35OSY: Operační systémy

Trendy a zajímavé koncepty v OS

Michal Sojka
michal.sojka@cvut.cz

4. leden, 2018

Obsah

1. Spolehlivost/bezpečnost
   - Windows
   - Qubes OS
   - Mikrojádra
     - GNU/Hurd
     - NOVA
   - Plan 9
2. Real-time OS
3. Open source
   - Ekonomické aspekty
   - Základy autorského práva a licence
4. Mobilní OS

Obsah

1 Spolehlivost/bezpečnost
- Windows
- Qubes OS
- Mikrojádra
  - GNU/Hurd
  - NOVA
- Plan 9

2 Real-time OS

3 Open source
- Ekonomické aspekty
- Základy autorského práva a licence

4 Mobilní OS

Problémy běžných OS

- Monolitický design
  - Jádro!
  - Uživatelské komponenty (X-server, . . .)
- Chyba v jedné části OS (např. ovladač zvukové karty) způsobí pád celého systému
- Skrze zranitelnost v jedné části OS lze napadnout jakoukoli jinou část OS
- Řešení
  - Lepší izolace jednotlivých komponent OS
  - Přesun komponent z jádra OS do uživatelského prostoru

Ovladače v uživatelském prostoru

- Chyba v ovladači může způsobit pád systému
- Nekvalitní ovladače jsou také zdrojem mnoha bezpečnostních problémů
- Ovladače v uživatelském prostoru:
  - Podporovány jak Linuxem (UIO) tak Windows
  - Spouštěny jako běžná aplikace
  - Přístup k registrům HW: mmap()
  - Obsluha přerušení – OS upozorní aplikaci pokud nastalo přerušení
    - UIO:

```
int uio = open("/dev/uio0", ...);
read(uio, ...); // waits for interrupt
handle_interrupt();
```

    - Při chybě ovladače ho lze jednoduše restartovat
    - Ostatní aplikace nevolají ovladač pomocí systémových volání, ale pomocí meziprocesní komunikace (např. fronty zpráv)
- OS založené na mikrojádře mají (téměř) všechny ovladače v uživatelském prostoru

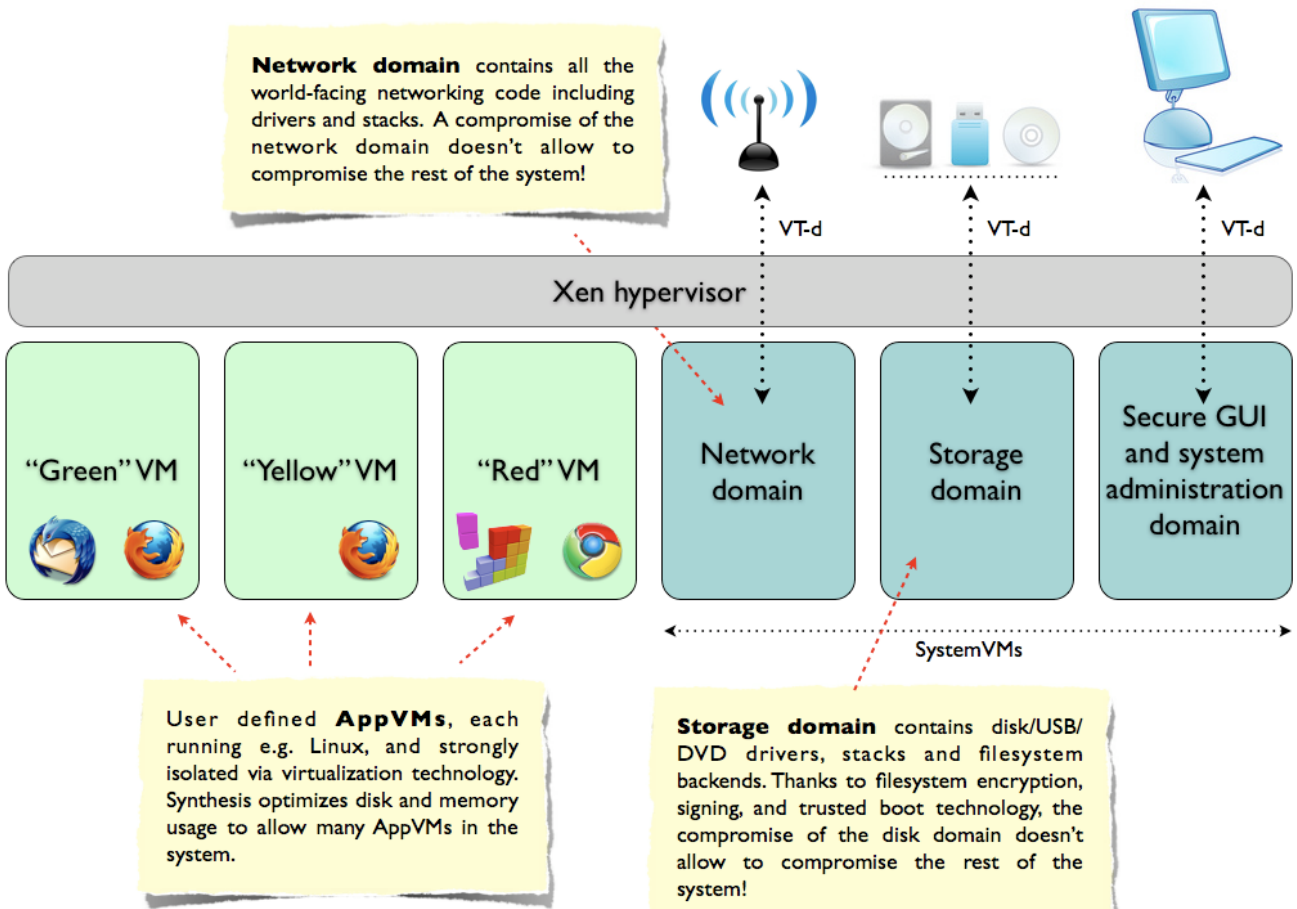
Windows

- Počínaje Windows NT 4.0 (1996), GUI bylo implementováno v jádře
  - převážně kvůli rychlosti $\Rightarrow$ nižší stabilita
- Počínaje Windows Vista (2006) a zejména Windows 7 (2009) byly některé komponenty přesunuty do uživatelského módu
  - Správce oken (DWM)
  - Zvukový server a související služby
  - Některé ovladače

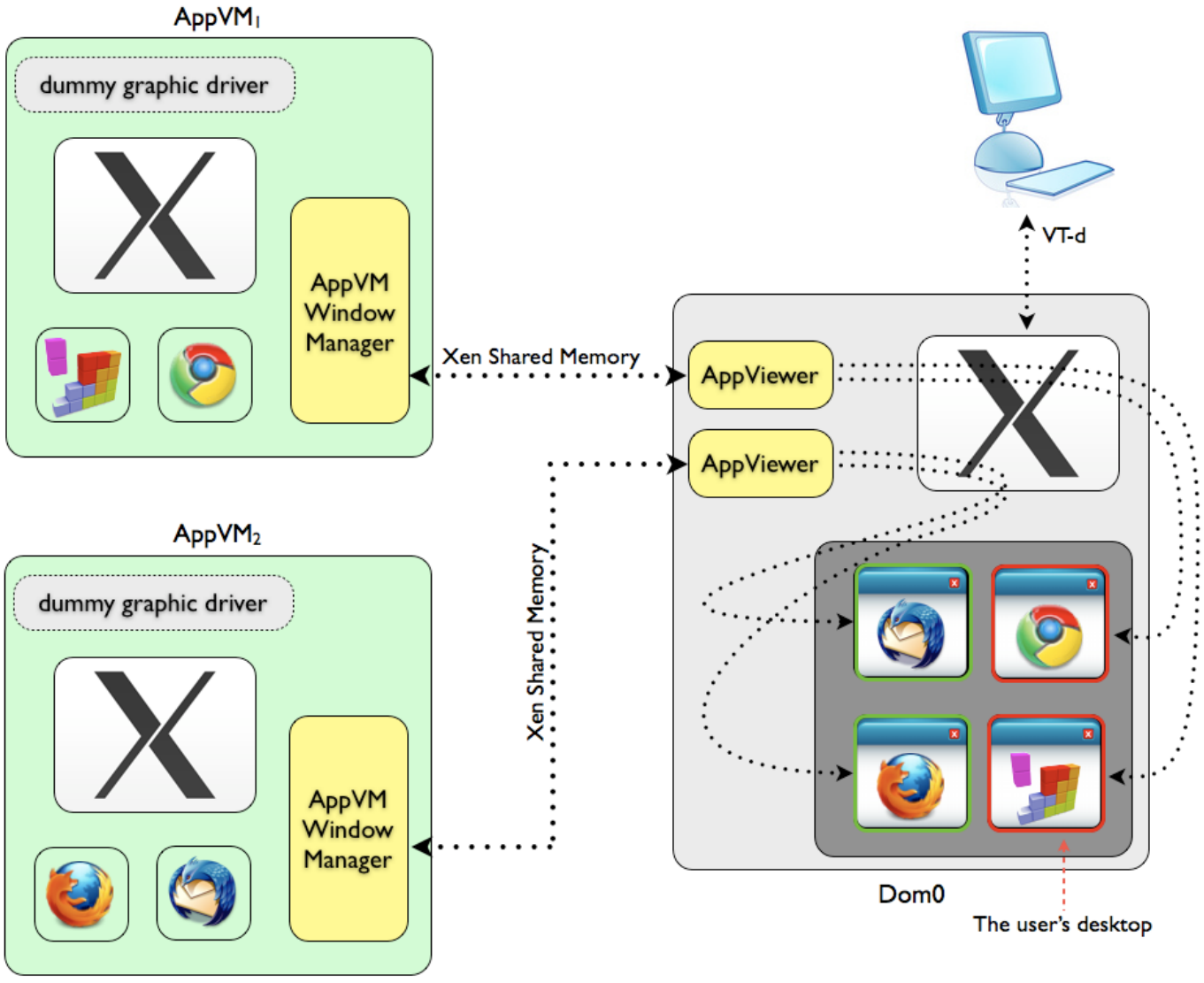
Qubes OS

https://www.qubes-os.org/

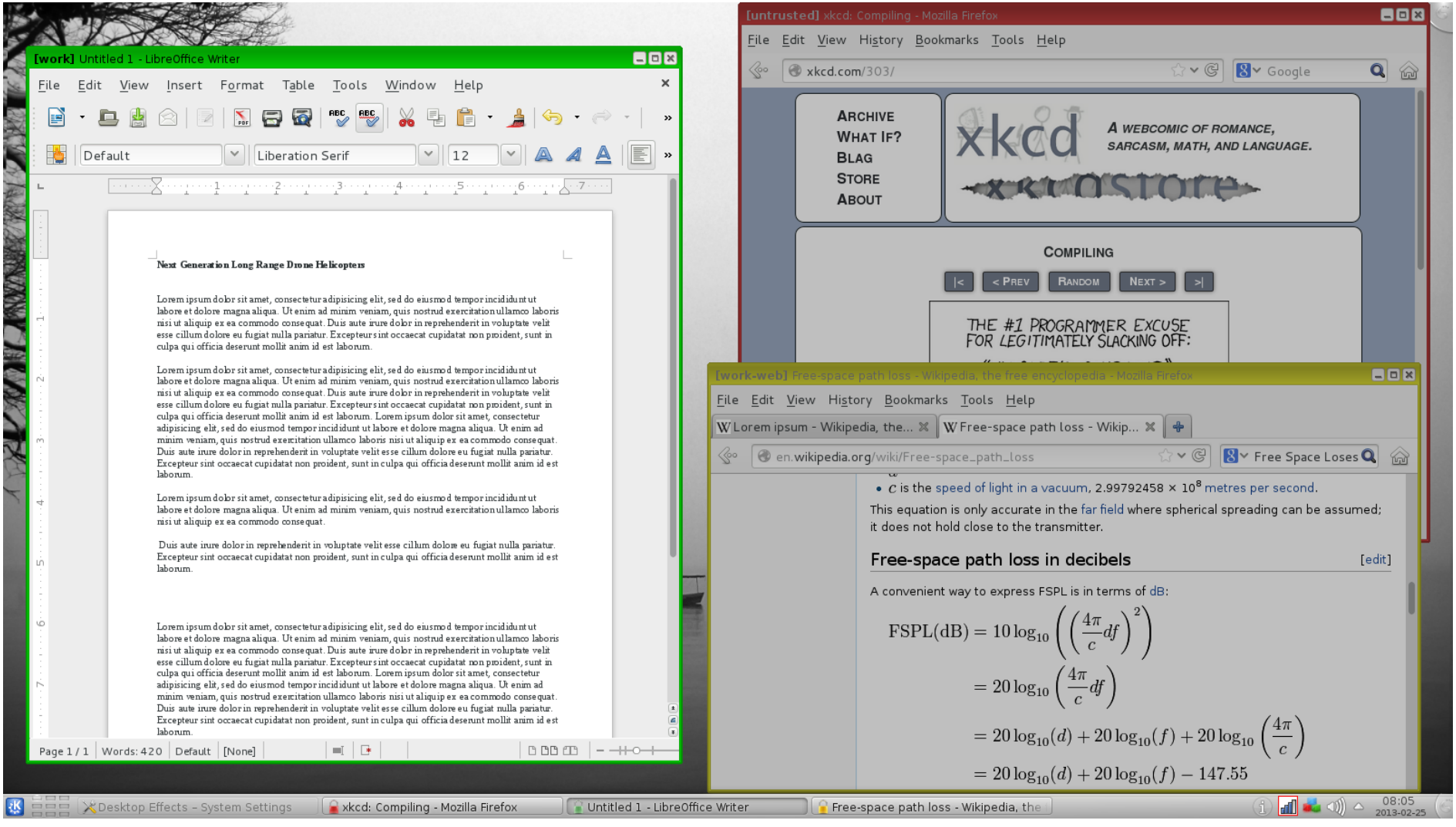
- Praktický a velmi bezpečný OS – základem je hypervizor Xen
- Jednotlivé komponenty OS běží jako virtuální stroje a komunikuji pomocí hypervizoru



Bezpečné GUI

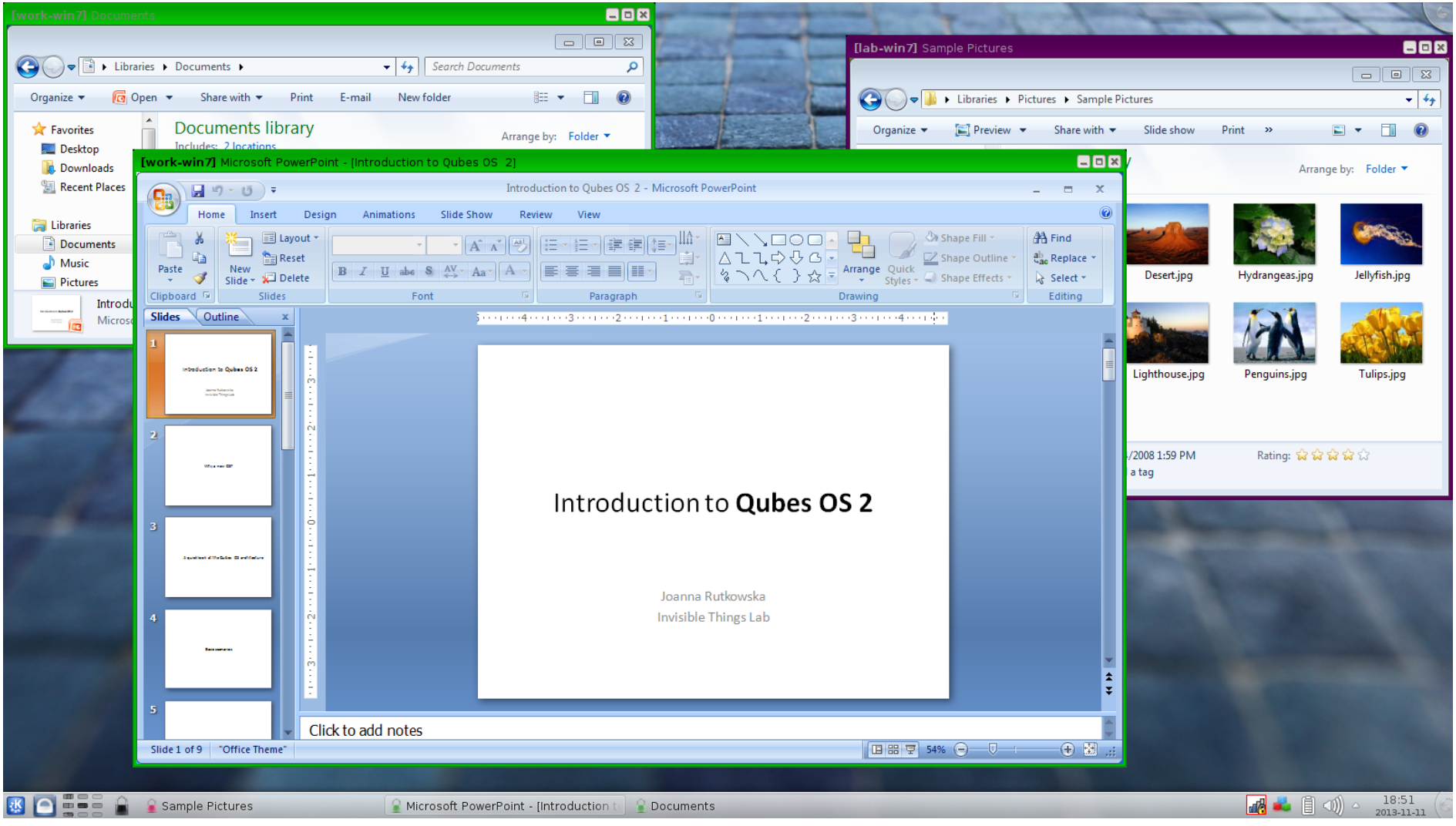


Qubes GUI



- Barevné označení oken podle zdrojové VM

Qubes GUI



- Integrace MS Windows

Mikrojádra

- V privilegovaném režimu procesoru implementují jen to, co nejde udělat jinde
- Jaká funkcionalita to je?

Mikrojádra

- V privilegovaném režimu procesoru implementují jen to, co nejde udělat jinde
- Jaká funkcionalita to je?
- Typicky
  - Správa adresních prostorů (procesů resp. stránkovacích tabulek)
  - Komunikace mezi vlákny/procesy pomocí zasílání zpráv
  - Plánování a synchronizace běhu vláken

Mikrojádra

- V privilegovaném režimu procesoru implementují jen to, co nejde udělat jinde
- Jaká funkcionalita to je?
- Typicky
  - Správa adresních prostorů (procesů resp. stránkovacích tabulek)
  - Komunikace mezi vlákny/procesy pomocí zasílání zpráv
  - Plánování a synchronizace běhu vláken
- V už. prostoru: vše ostatní
  - správa paměti (alokátor)
  - souborové systémy
  - síťové protokoly
  - ovladače

GNU/Hurd

https://www.gnu.org/software/hurd/

- Cíl: Nahradit monolitická jádra používaná s OS GNU mikrojádrem
- OS GNU

GNU/Hurd

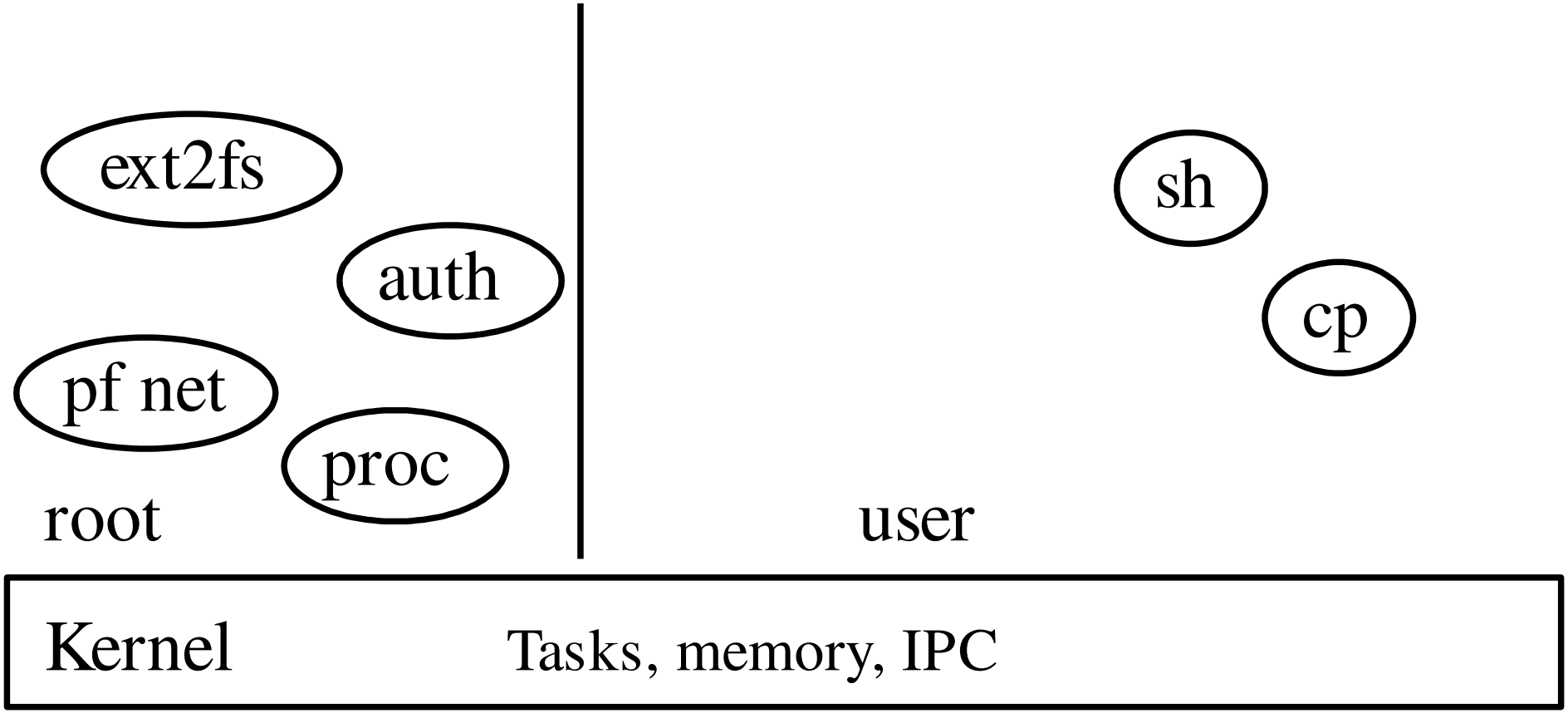
https://www.gnu.org/software/hurd/

- Cíl: Nahradit monolitická jádra používaná s OS GNU mikrojádrem
- OS GNU
  - Svobodný OS Unixového typu
  - Založen v r. 1984 R. M. Stallmanem
  - Dnes používán převážně s jádrem Linux
- Hurd ve vývoji od r. 1990
- Funkční, ale mnoho problémů (ovladače, . . .)
- Lze si stáhnout image a experimentovat např. v Qemu
- Distribuce GNU/Debian je portována na Hurd

GNU/Hurd

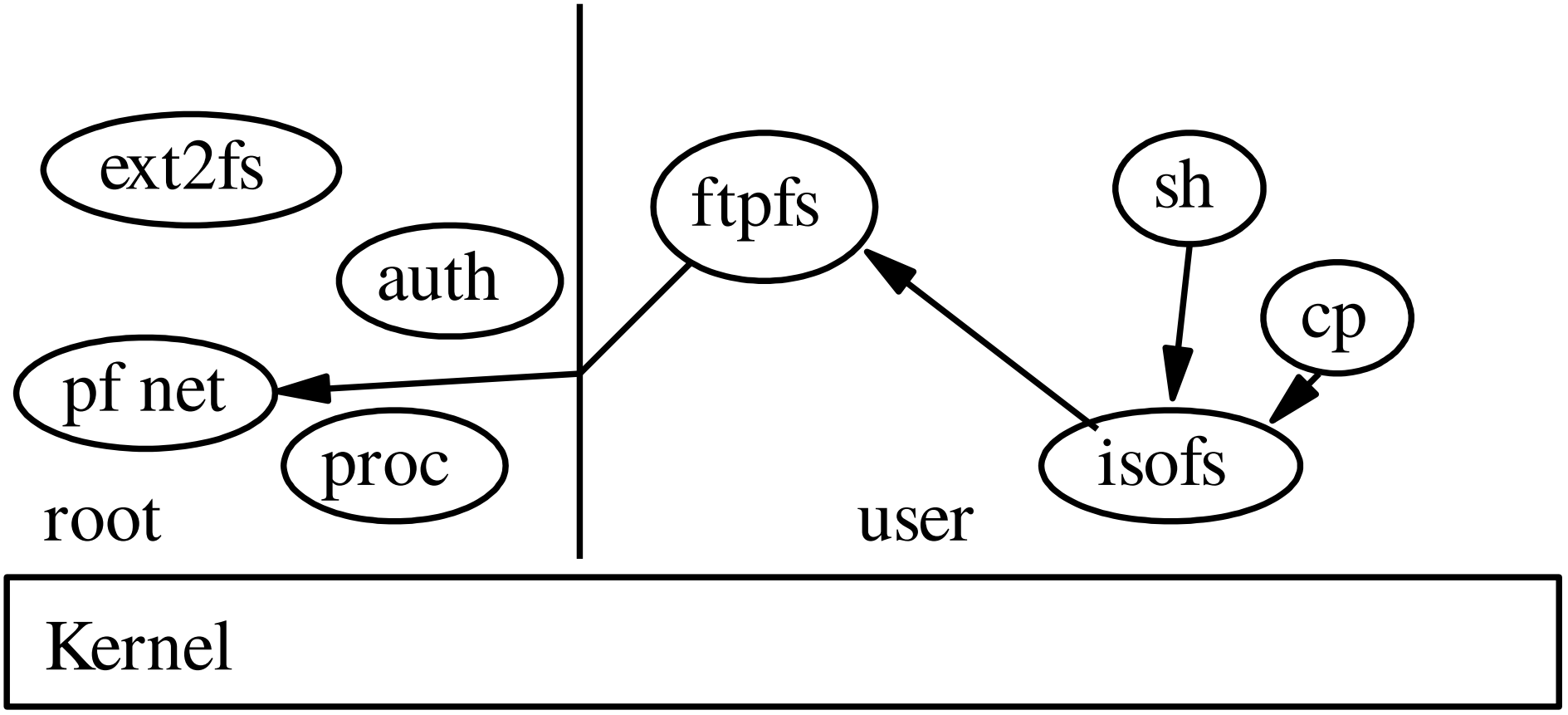
Základní struktura

Hurd = jádro Mach a uživatelské servery poskytující služby jako např. síť, souborový systém, . . .



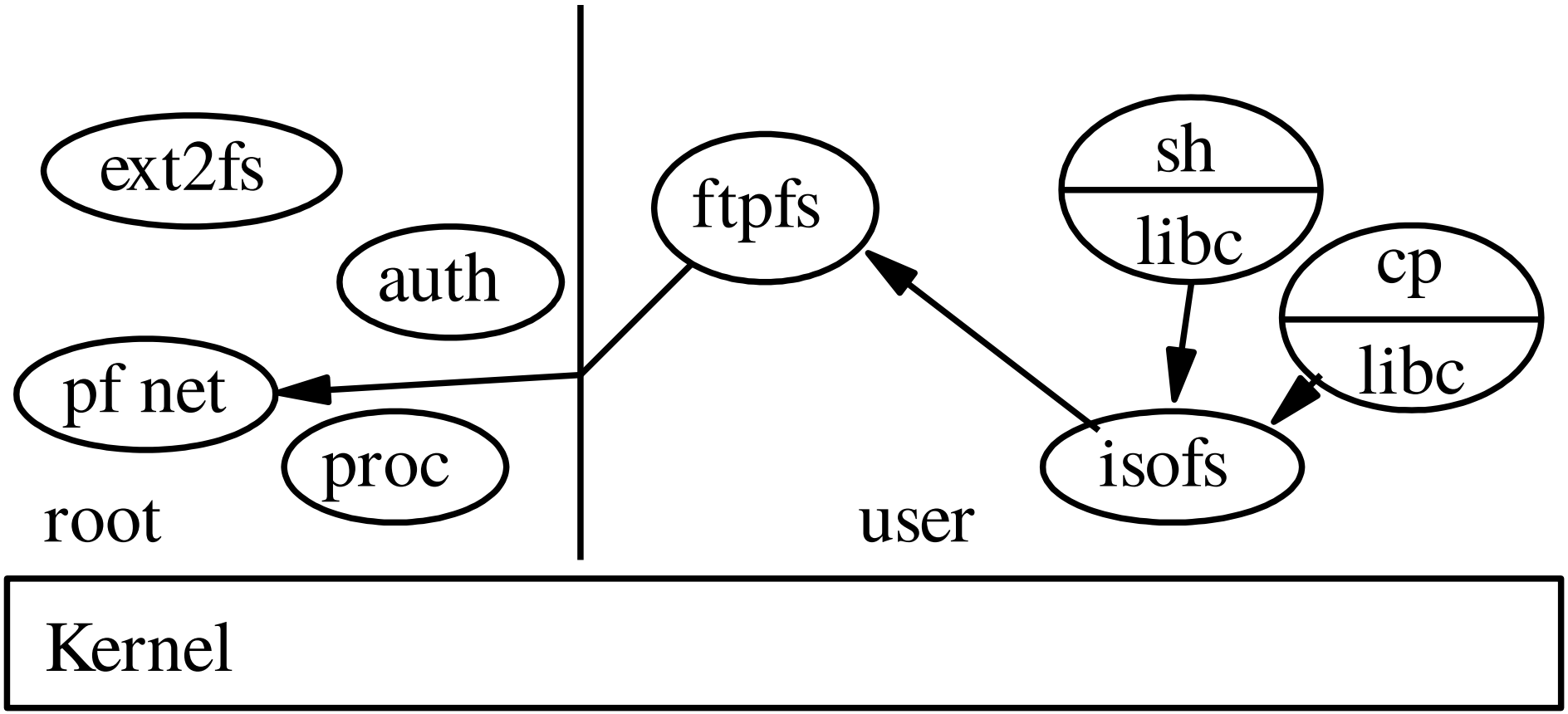
GNU/Hurd

Souborový systém je poskytován aplikacím serverem. Zde transparentní přístup k souborům v ISO obrazu na vzdáleném serveru přístupném protokolem FTP.



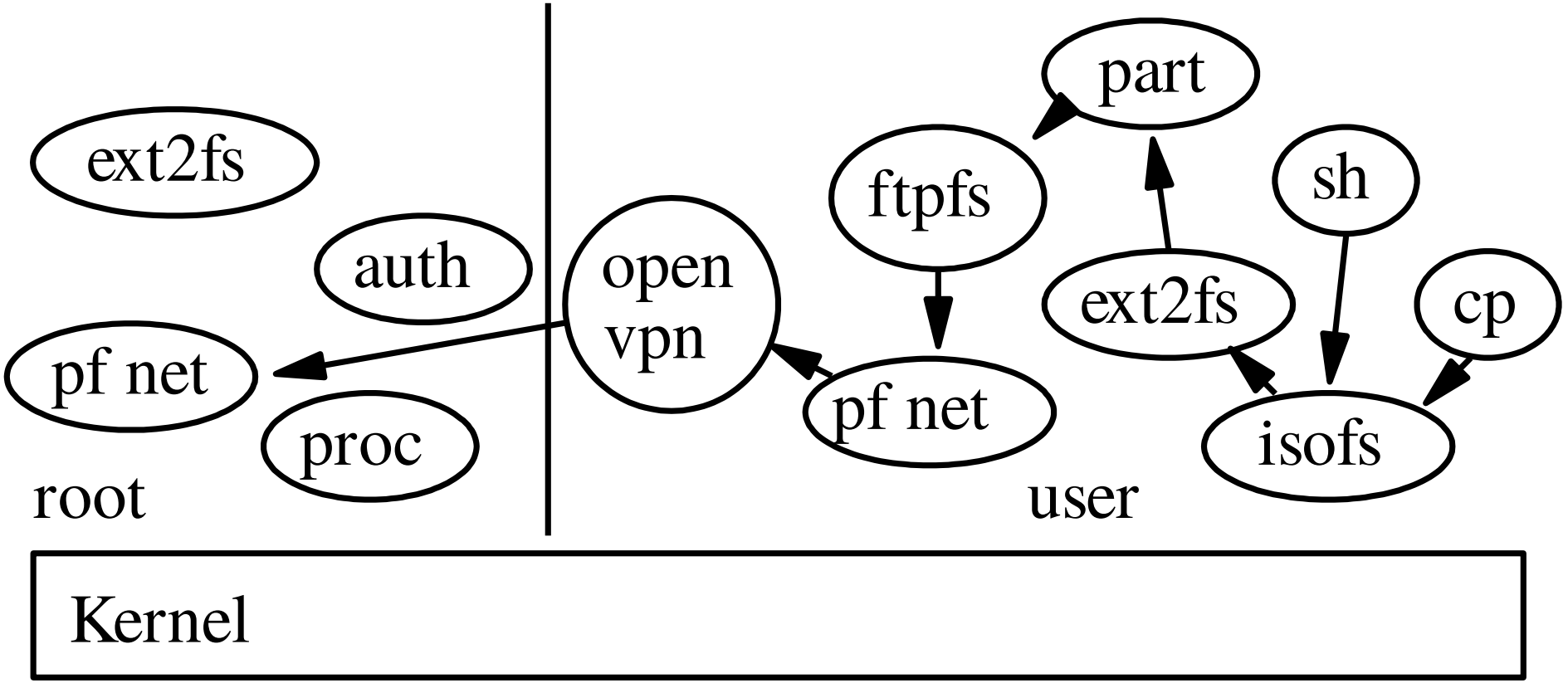
GNU/Hurd

Tradiční systémové služby jsou nejsou dostupné přes systémová volání, ale přes vzdálené volání procedur (RPC)



GNU/Hurd

Propojení serverů lze libovolně kombinovat. ISO image na diskovém oddílu na FTP přístupném přes VPN.



NOVA

http://hypervisor.org/

- Výkonný „mikrohypervizor“ (9000 řádek kódu)
- Vývoj začal na TU Dresden
- Bezpečnost je řešena pomocí „schopností“ (capability, viz 8. přednáška)
- Přehledná specifikace – jen několik jednoduchých konceptů:

  `https://github.com/udosteinberg/NOVA/raw/master/doc/specification.pdf`

NOVA

http://hypervisor.org/

- Výkonný „mikrohypervizor“ (9000 řádek kódu)
- Vývoj začal na TU Dresden
- Bezpečnost je řešena pomocí „schopností“ (capability, viz 8. přednáška)
- Přehledná specifikace – jen několik jednoduchých konceptů:

  `https://github.com/udosteinberg/NOVA/raw/master/doc/specification.pdf`
- Objekty spravované jádrem:
  - Protection domain – částečně podobné procesům z jiných OS
  - Execution context } vlákno
  - Scheduling context } vlákno
  - Portal – něco mezi rourou a UNIXovým socketem
  - Semaphore

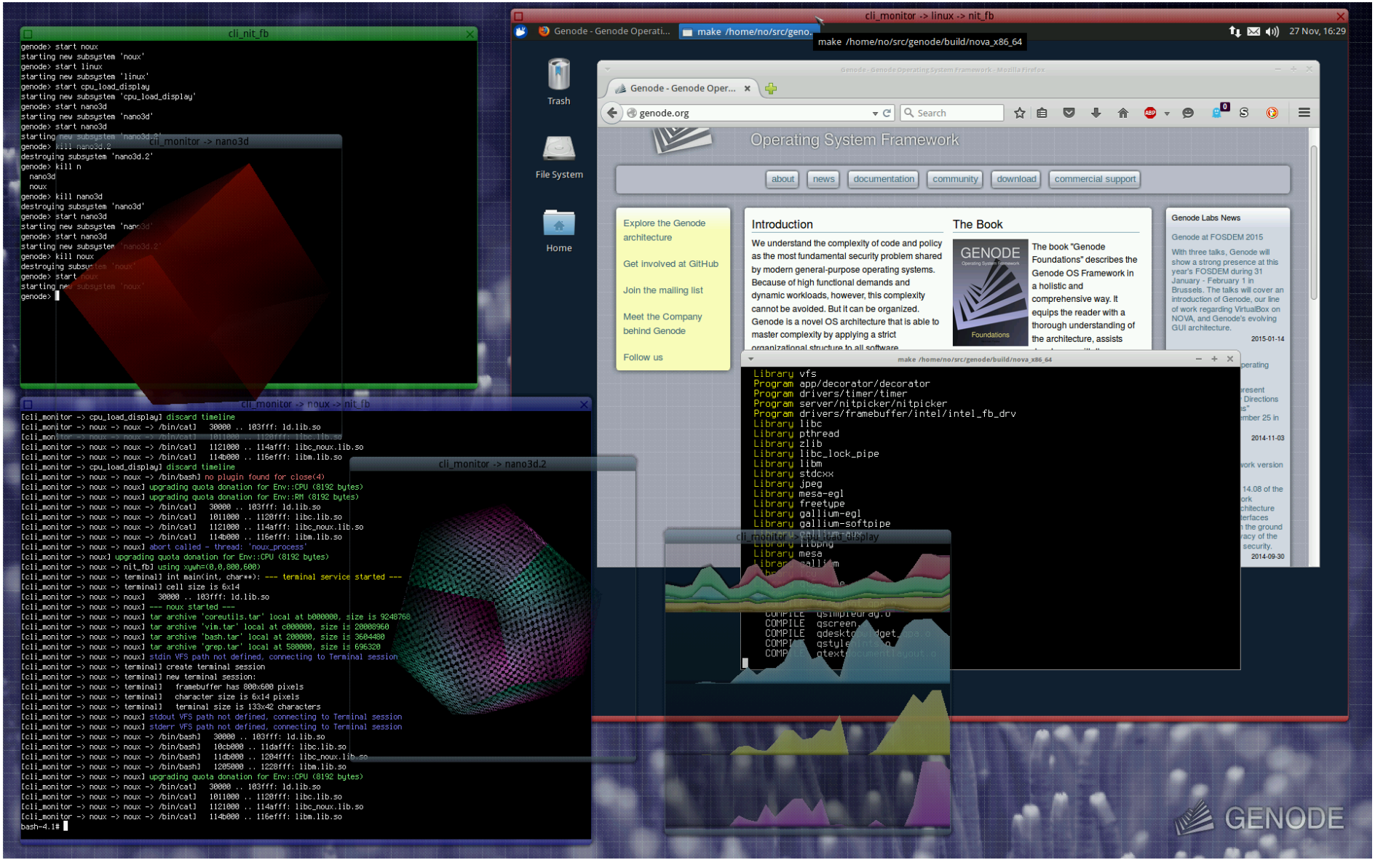
NOVA

http://hypervisor.org/

- Výkonný „mikrohypervizor“ (9000 řádek kódu)
- Vývoj začal na TU Dresden
- Bezpečnost je řešena pomocí „schopností“ (capability, viz 8. přednáška)
- Přehledná specifikace – jen několik jednoduchých konceptů:

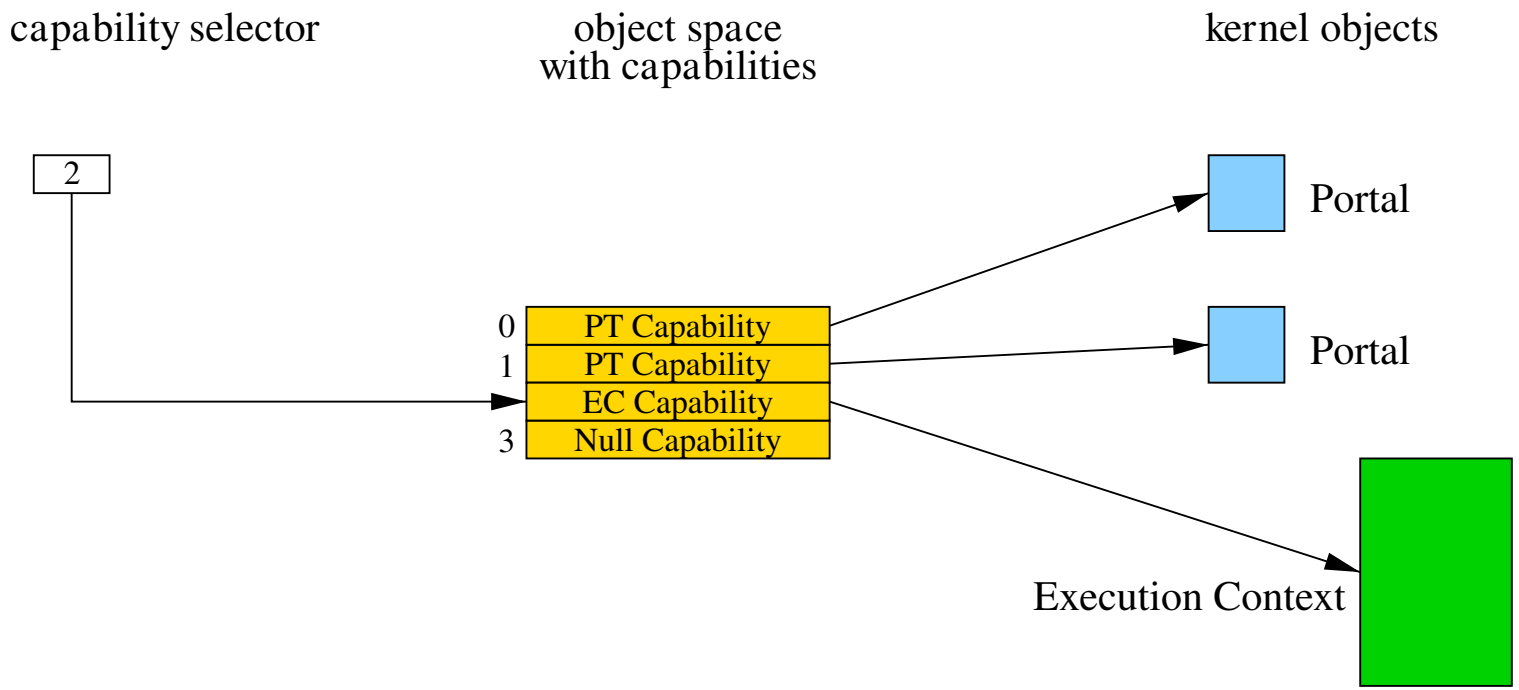
  `https://github.com/udosteinberg/NOVA/raw/master/doc/specification.pdf`
- Objekty spravované jádrem:
  - Protection domain – částečně podobné procesům z jiných OS
  - Execution context } vlákno
  - Scheduling context } vlákno
  - Portal – něco mezi rourou a UNIXovým socketem
  - Semaphore
- Uživatelské prostředí: např. `https://genode.org/`

Genode OS framework



Protection domain

- Objekt zodpovědný za izolaci
- Skládá se z:
  - paměťový prostor – stránky fyzické paměti
  - prostor I/O portů – přístupné I/O porty
  - prostor jaderných objektů – ostatní objekty
- S „prostory“ se pracuje pomocí schopností, které jsou indexovány selectory (celé číslo – int)



Vlákna

- To, co jiné OS nazývají vlákny jsou v OS NOVA dva různé objekty
- **Execution context** (EC) reprezentuje stav procesoru daného „vlákna“ a obsahuje
  - Uložené registry procesoru
  - Odkaz na „protection domain“
  - …
- **Scheduling context** (SC) – informace pro rozvrhovač
  - Priorita
  - Časové kvantum
  - Odkaz na execution context
- Koncepty jsou oddělené proto, že když nějaký „server“ (reprezentovaný EC) vykonává službu pro jiný EC (např. aplikaci), propůjčí aplikace svůj SC serverovému EC.

Systémová volání jádra NOVA

1. **Call** – komunikace pomocí portálů a „delegování“ schopností
2. **Reply** – odpověď na call a „delegování“ schopností zpět
3. **Create PD**
4. **Create EC**
5. **Create PT**
6. **Create SC**
7. **Create SM**
8. **Revoke** – odnímání delegovaných schopností
9. **Lookup** – zjišťování stavu/existence schopností
10. **EC Ctrl** – podobné zaslání signálu jinému EC
11. **SC Ctrl** – zjištění doby běhu
12. **PT Ctrl**
13. **SM Ctrl** – operace nad semaforem
14. **Assign PCI** – delegování přístupu k PCI zařízení
15. **Assign GSI** – nastavení routování přerušení

Příklad vytvoření „procesu“ v OS NOVA

Zjednodušený pseudokód

```
unsigned cap = 2048; // start of block of free capabilities
unsigned cap_newec = cap+0, cap_newpd = cap+1, cap_newsc = cap+2;
unsigned cap_nameserver_pt = cap+3;
unsigned event_pt_base = 32;
// Vytvor komunikacni kanaly pro "potomka"
// Kdyz potomek zavola "call" na danem portalu, vyvola nase funkce
create_pt(cap_nameserver_pt, cap_self_pd, ..., nameserver_handler);
create_pt(cap + event_pt_base + STARTUP_EVT, ..., child_startup);
create_pt(cap + event_pt_base + PAGE_FAULT_EVT, ..., child_page_fault);
// Vytvor PD se zakladnimi schopnostmi a hlavni "vlakno"
create_pd(cap_newpd, cap_self_pd, cap_range(cap, cap+256));
create_ec(cap_newec, cap_newpd, CPU_0, UTCB, stack_ptr, event_pt_base);
create_sc(cap_newsc, cap_newpd, cap_newec)
```

Plan 9

https://9p.io/plan9/

- Vyvíjen v Bellových laboratořích – z části i původními autory UNIXu
- Dotažení UNIXové filozofie „Všechno je soubor“ do „úplné dokonalosti“
- Distribuovaný OS
- Na rozdíl od UNIXu integruje síť do konceptu OS – systém může běžet na více počítačích dohromady

Obsah

1 Spolehlivost/bezpečnost
- Windows
- Qubes OS
- Mikrojádra
  - GNU/Hurd
  - NOVA
- Plan 9

2 Real-time OS

3 Open source
- Ekonomické aspekty
- Základy autorského práva a licence

4 Mobilní OS

Operační systémy reálného času

- Pokud počítač interaguje s objekty v reálném světě, je často potřeba, aby kromě běžných požadavků na OS navíc garantoval dodržení časových parametrů
  - Např. zpoždění obsluhy přerušení
- Běžný OS takově časování negarantuje a při větším zatížení se může „zasekávat“
- Používají se real-time OS (RTOS)
- Typické aplikace:
  - Řízení letadel
  - Průmysl (např. obráběcí CNC stroje)
  - Robotika
- Více v předmětu x35PSR

Bezpečnostně-kritické aplikace

- Aplikace interagující s reálným světem mají často „bezpečnostně-kritický“ charakter
- Selhání takové aplikace může mít katastrofální následky
  - Výbuch atomové elektrárny
  - Pád letadla
  - Srážka vlaků
- Bezpečnostně-kritické aplikace
  - spouštějí se na spolehlivém (tj. pomalém) HW
  - jsou co nejjednodušší, aby šly dobře verifikovat
- Ideální pro nasazení mikrojader (např. PikeOS)

Bezpečnostně-kritické aplikace

- Aplikace interagující s reálným světem mají často „bezpečnostně-kritický“ charakter
- Selhání takové aplikace může mít katastrofální následky
  - Výbuch atomové elektrárny
  - Pád letadla
  - Srážka vlaků
- Bezpečnostně-kritické aplikace
  - spouštějí se na spolehlivém (tj. pomalém) HW
  - jsou co nejjednodušší, aby šly dobře verifikovat
- Ideální pro nasazení mikrojader (např. PikeOS)
- Častým požadavkem na bezp.-kritické aplikace je tzv. freedom from interference (nepodléhání rušení)
  - Aplikace se skládá z komponent
  - Při vývoji bychom chtěli testovat komponenty samostatně a mít jistotu, že po jejich integraci budou fungovat stejně
  - Může to fungovat, pokud nám někdo (např. RTOS) garantuje, že samostatně běžící komponenta poběží stejně jako při běhu s jinými komponentami
  - Této vlastnosti se říká temportal isolation (časová izolace)

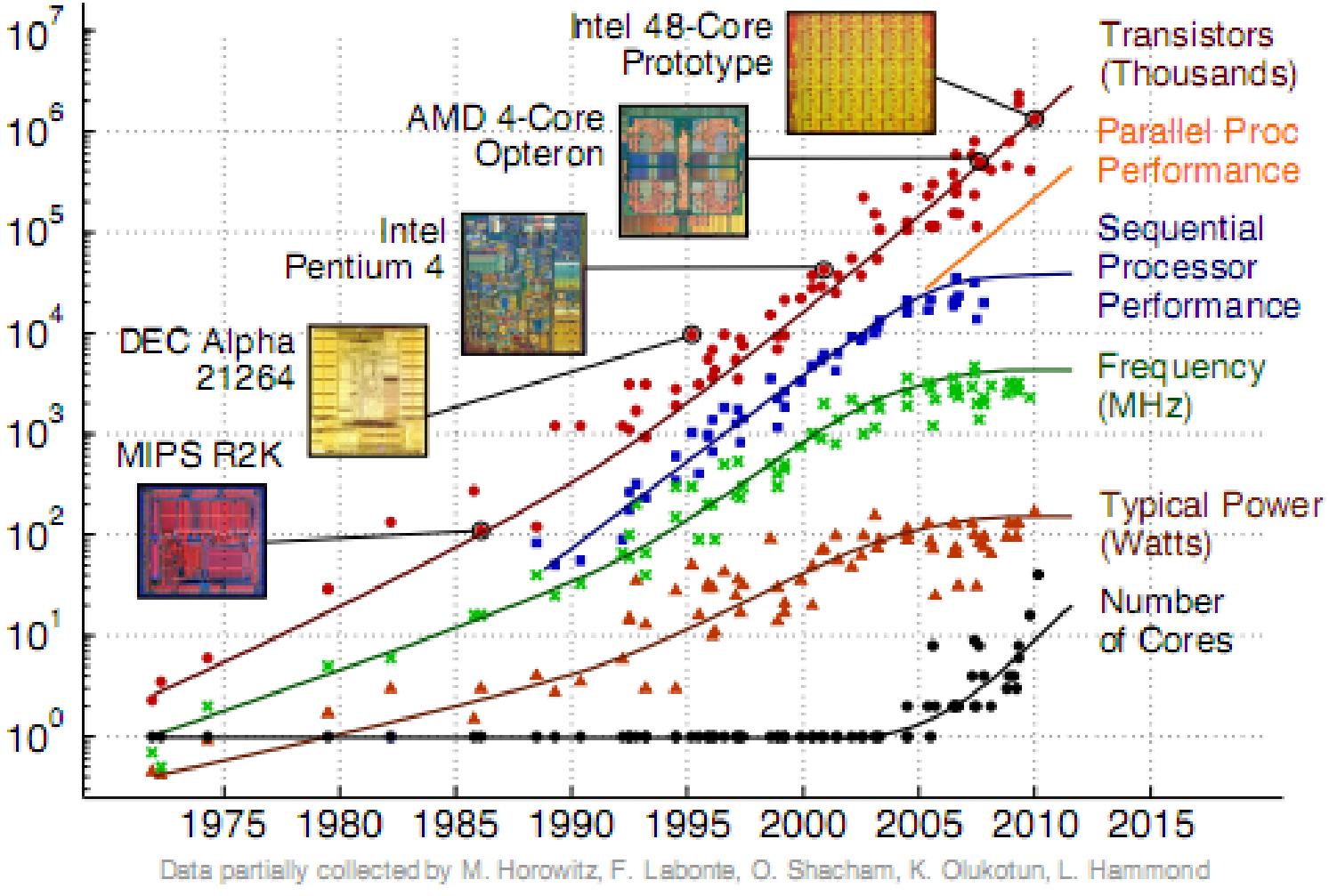
Bezpečnostně-kritické aplikace budoucnosti

- Autonomně řízená auta
  - Pro zpracování dat ze senzorů je potřeba velký výpočetní výkon
  - Aplikace jsou složité (umělá inteligence, zpracování obrazu, . . .)
- Pravý opak dnešních bezpečnostně-kritických aplikací
- U takových aplikací je velmi těžké zajistit „freedom from interence“

Bezpečnostně-kritické aplikace budoucnosti

- Autonomně řízená auta
  - Pro zpracování dat ze senzorů je potřeba velký výpočetní výkon
  - Aplikace jsou složité (umělá inteligence, zpracování obrazu, . . .)
- Pravý opak dnešních bezpečnostně-kritických aplikací
- U takových aplikací je velmi těžké zajistit „freedom from interence“
- Proč?

Many-core revolution



Prepared by C. Batten - School of Electrical and Computer Engineering - Cornell University - 2005 - retrieved Dec 12 2012 - http://www.csl.cornell.edu/courses/ece5950/handouts/ece5950-overview.pdf

Vícejádrové CPU a heterogenní SoC

- Díky fyzikálním omezením dnes nelze zvyšovat výpočetní výkon zvyšováním taktovací frekvence CPU
- Zvyšuje se tedy počet CPU na čipu
- CPU ale nejsou 100% nezávislá – sdílí například paměť

Vícejádrové CPU a heterogenní SoC

- Díky fyzikálním omezením dnes nelze zvyšovat výpočetní výkon zvyšováním taktovací frekvence CPU
- Zvyšuje se tedy počet CPU na čipu
- CPU ale nejsou 100% nezávislá – sdílí například paměť
- Pokud mám 2jádrové CPU, výpočet, který nepotřebuje moc paměti (např. hledání prvočísel) běží 2$\times$ rychleji

Vícejádrové CPU a heterogenní SoC

- Díky fyzikálním omezením dnes nelze zvyšovat výpočetní výkon zvyšováním taktovací frekvence CPU
- Zvyšuje se tedy počet CPU na čipu
- CPU ale nejsou 100% nezávislá – sdílí například paměť
- Pokud mám 2jádrové CPU, výpočet, který nepotřebuje moc paměti (např. hledání prvočísel) běží 2$\times$ rychleji
- Výpočty, které přistupují k paměti jsou ale zpomalovány ostatními jádry – na dnešních CPU lze pozorovat až 5násobné zpomalení

Vícejádrové CPU a heterogenní SoC

- Díky fyzikálním omezením dnes nelze zvyšovat výpočetní výkon zvyšováním taktovací frekvence CPU
- Zvyšuje se tedy počet CPU na čipu
- CPU ale nejsou 100% nezávislá – sdílí například paměť
- Pokud mám 2jádrové CPU, výpočet, který nepotřebuje moc paměti (např. hledání prvočísel) běží 2× rychleji
- Výpočty, které přistupují k paměti jsou ale zpomalovány ostatními jádry – na dnešních CPU lze pozorovat až 5násobné zpomalení
- Ještě horší je situace u heterogenního HW: CPU + paralelní akcelerátor (GPU/FPGA)
  - GPU má mnohem větší šířku pásma k přístupu do paměti a „zpomaluje“ CPU mnohem víc
- Freedom from interference? Těžko

Řešení

1. Vývoj speciálního HW bez sdílených komponent $\Rightarrow$ drahé
2. Aktuální výzkum v oblasti RTOS: Predictable Execution Model
   - Speciální způsob kompilace zdrojového kódu
   - Shlukování přístupů do paměti a speciální plánování vláken, aby nedocházelo k interferenci
   - Naše výsledky: 9násobné snížení interference mezi jádry na platformě NVIDIA Drive CX2

Závody F1/10 s univerzitou z italské Modeny

Zájem médií

Budoucnost

- Italové spolupracují s Maserati
- ČVUT s Porsche
- Na léto je plánován podobný „závod“ Maserati vs. Porsche na italském autodromu

Obsah

1 Spolehlivost/bezpečnost
- Windows
- Qubes OS
- Mikrojádra
  - GNU/Hurd
  - NOVA
- Plan 9

2 Real-time OS

3 Open source
- Ekonomické aspekty
- Základy autorského práva a licence

4 Mobilní OS

Historie FOSS

- FOSS = Free and Open Source Software
- 50 a 60 léta: open source (public domain) bylo normou
  - Lidé mezi sebou sdíleli kód jako kuchařské recepty nebo znalosti matematiky
- V 70. letech začaly firmy software „uzavírat“ a prodávat (AT&T Unix, Microsoft, . . .)
- Richard M. Stallman (RMS)
  - 1980 – naštvala ho nemožnost opravit chybu v softwaru nové tiskárny na MIT
  - 1983 – začal vyvíjet operační systém GNU („GNU’s not Unix“)

Svobodný software

Program je svobodný software, pokud uživatelé toho programu mají čtyři základní svobody:

- Svoboda 0: spuštět program k libovolnému účelu
- Svoboda 1: studovat jak program funguje a měnit ho
- Svoboda 2: redistribuovat kopie programu
- Svoboda 3: vylepšovat program a zveřejňovat svá zlepšení

— Richard Stallman, 1986 (original version)

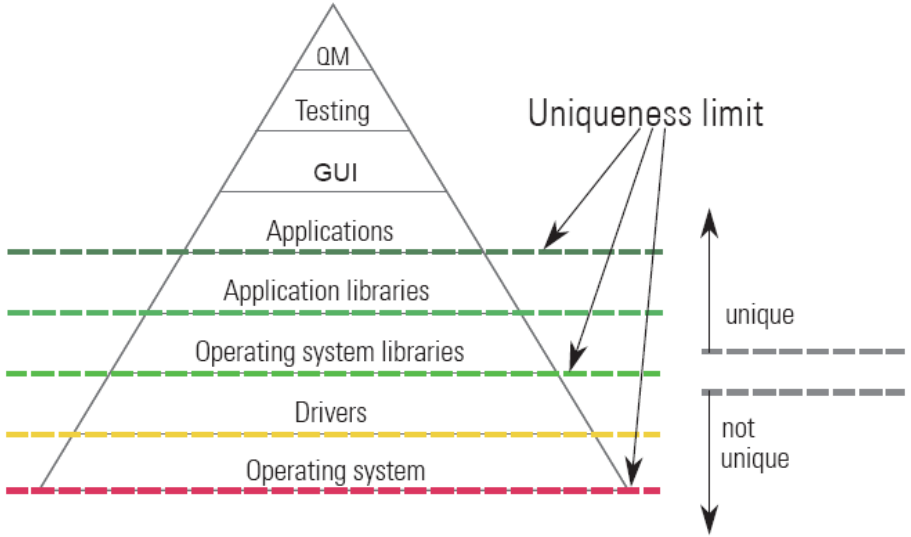
http://www.gnu.org/philosophy/free-sw.en.html

Open source

- Termín vznikl okolo roku 1998
- Označuje otevřený vývojový model a snaží se odlišit od filozoficky a politicky motivovaného pojmu „svobodný software“
- Definice open source: `https://opensource.org/osd-annotated`

Otevřít nebo neotevřít?

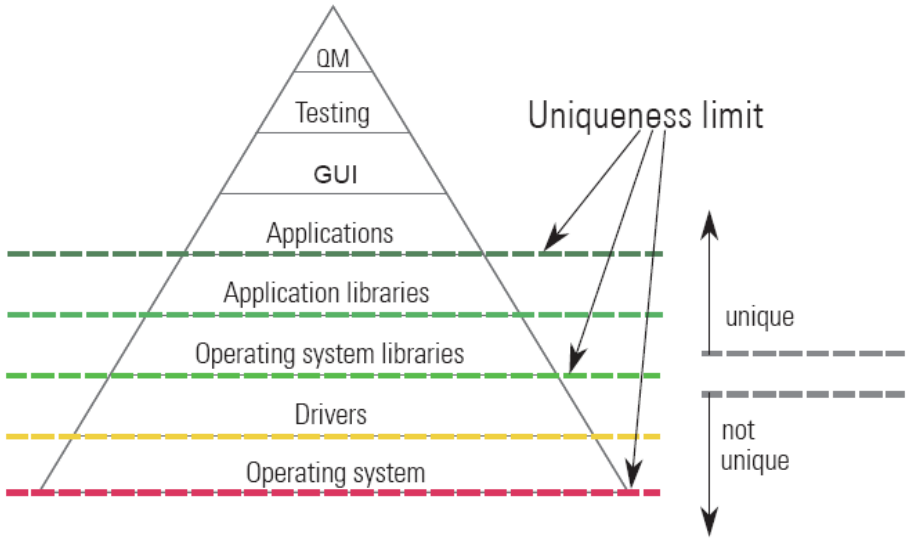
- Firemní know-how:
  - konkurenční výhoda (malá část)
  - ostatní (velká část)
- Maximalizace ekonomického úspěchu = maximalizace investic do konkurenční výhody
- Ostatní know-how
  - Koupit ho
  - Spolupráce s ostatními (i s konkurenty) $\rightarrow$ FOSS
    - Standardizace



Zdroj: OSADL

Otevřít nebo neotevřít?

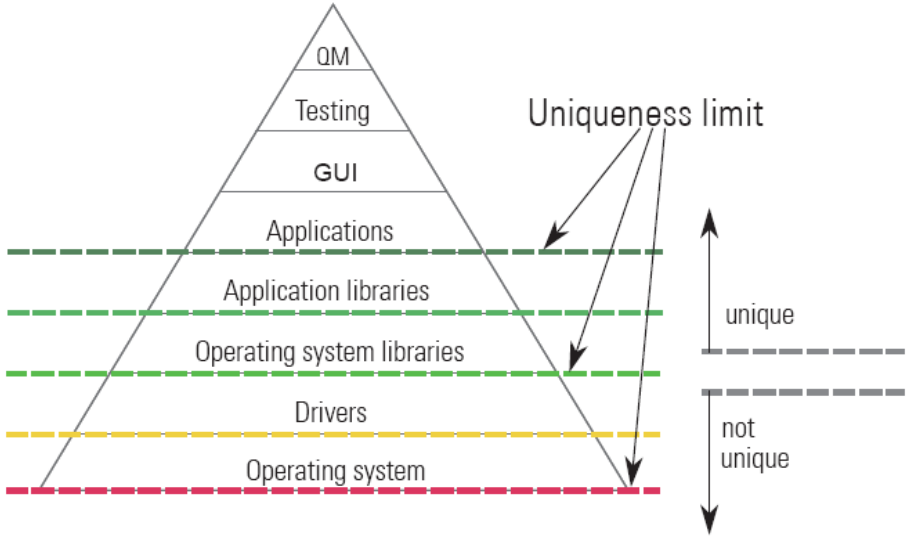
- Firemní know-how:
  - konkurenční výhoda (malá část)
  - ostatní (velká část)
- Maximalizace ekonomického úspěchu = maximalizace investic do konkurenční výhody
- Ostatní know-how
  - Koupit ho
  - Spolupráce s ostatními (i s konkurenty) $\rightarrow$ FOSS
    - Standardizace
- Globalizace
  - Tlak na firmy – více konkurence, levnější práce jinde



Zdroj: OSADL

Otevřít nebo neotevřít?

- Firemní know-how:
  - konkurenční výhoda (malá část)
  - ostatní (velká část)
- Maximalizace ekonomického úspěchu = maximalizace investic do konkurenční výhody
- Ostatní know-how
  - Koupit ho
  - Spolupráce s ostatními (i s konkurenty) $\rightarrow$ FOSS
    - Standardizace
- Globalizace
  - Tlak na firmy – více konkurence, levnější práce jinde
  - Umožňuje vývoj FOSS
  - FOSS lze chápat jako kompenzace negativních efektů globalizace



Zdroj: OSADL

Jak vydělat peníze s FOSS?

Mnoho způsobů – nejběžnější:

1. **Prodej HW:** Dostupnost FOSS zvyšuje užitečnost/prodej HW – Android, Intel, . . .

Jak vydělat peníze s FOSS?

Mnoho způsobů – nejběžnější:

1. **Prodej HW:** Dostupnost FOSS zvyšuje užitečnost/prodej HW – Android, Intel, ...
2. **Placená podpora:** SW je zdarma; lidé platí za vývojářovo know-how (konzultace) nebo vývoj rozšíření SW (také FOSS) – Cygnus = *Cygnus, your GNU Support*, RedHat

Jak vydělat peníze s FOSS?

Mnoho způsobů – nejběžnější:

1. **Prodej HW:** Dostupnost FOSS zvyšuje užitečnost/prodej HW
   – Android, Intel, . . .
2. **Placená podpora:** SW je zdarma; lidé platí za vývojářovo know-how (konzultace) nebo vývoj rozšíření SW (také FOSS)
   – Cygnus = *Cygnus, your GNU Support*, RedHat
3. **Dvojité licencování**: copyleftová licence nebo placená komerční licence
   – Qt, MySQL

Jak vydělat peníze s FOSS?

Mnoho způsobů – nejběžnější:

1. **Prodej HW:** Dostupnost FOSS zvyšuje užitečnost/prodej HW
   – Android, Intel, . . .
2. **Placená podpora:** SW je zdarma; lidé platí za vývojářovo know-how (konzultace) nebo vývoj rozšíření SW (také FOSS)
   – Cygnus = *Cygnus, your GNU Support*, RedHat
3. **Dvojité licencování**: copyleftová licence nebo placená komerční licence
   – Qt, MySQL
4. **Reklama:** FOSS dělá reklamu jinému produktu/firmě a je jím placen
   – Firefox, (Ubuntu)

Jak vydělat peníze s FOSS?

Mnoho způsobů – nejběžnější:

1. **Prodej HW:** Dostupnost FOSS zvyšuje užitečnost/prodej HW
   – Android, Intel, . . .
2. **Placená podpora:** SW je zdarma; lidé platí za vývojářovo know-how (konzultace) nebo vývoj rozšíření SW (také FOSS)
   – Cygnus = *Cygnus, your GNU Support*, RedHat
3. **Dvojité licencování**: copyleftová licence nebo placená komerční licence
   – Qt, MySQL
4. **Reklama:** FOSS dělá reklamu jinému produktu/firmě a je jím placen
   – Firefox, (Ubuntu)
5. **Placená rozšíření:**
   – Eclipse IDE

Duševní vlastnictví

- Duševní vlastnictví jsou výsledky procesu lidské tvořivosti, zkoumání a myšlení. Těmi jsou myšleny různé výsledky více či méně originálních myšlenek, námětů, návodů a řešení.
- Lze jej chránit pomocí:
  - patentů
  - ochranných známek
  - zákonů o obchodním tajemství
  - autorského práva (copyright)
- Software spadá pod duševní vlastnictví

Autorské právo

- Omezuje použití specifických vyjádření myšlenek
- t.j., může být použito k omezení činností jako např.:
  - Tvorba **kopií** díla a jejich prodej
  - Tvorba **odvozených** děl
  - **Veřejné provozování** díla
  - **Prodej** a převod autorských práv na jiné osoby
- Vztahuje se na cokoli, co vykazuje známky kreativity
- Automaticky se vztahuje na všechno co vytvoříte, pokud se to vyskytuje v nějaké pevné a konkrétní podobě
  - Výchozí nastavení $\approx$ autor má monopol, ostatní nemohou s dílem nakládat
- Omezení:
  - Trvání: obvykle 70–150 let (podle státu)
  - Férové použití: svoboda slova, svoboda citování atd. (v českých zákonech toto omezení není)
- Téměř shodné po celém světě – od Bernské úmluvy z roku 1886

Proč jsou potřeba licence?

- Autorské právo platí pro software
- Výchozí nastavení je „všechna práva vyhrazena“

Z pohledu uživatele

- Bez licence nemůžete se softwarem dělat skoro nic.

Z pohledu autora

- Bez licence nemohou vaší (potenciální) uživatele SW používat
- Musíte jim dát aspoň nějaká práva

Licence svobodného software a autorské právo

- Licence svobodného software jsou právní „hack“: jsou podobné ostatním licencím, ale místo toho, aby činnost uživatelů omezovaly, tak naopak uživatelům některá „speciální“ práva dávají
- Licence FOSS dávají uživatelům právě ta práva, aby si uživatelé **mohli užívat 4 základní svobody** (spouštět, studovat, kopírovat a modifikovat)
- Ale to neznamená, že je s FOSS možné dělat cokoli – FOSS většinou licence vyžadují plnění určitých **určitých podmínek**
  - Pokud je uživatel nedodrží, licence pro něj neplatí a tudíž platí výchozí nastavení autorského práva „všechna práva vyhrazena“.

Poznámka; FOSS **není proti ochraně duševního vlastnictví**. Ve skutečnosti licence FOSS využívají autorského práva k zajištění svobody SW.

Kategorie FOSS licencí

Licence FOSS mohou být **klasifikovány podle podmínek, které vyžadují** výměnou za svobodu softwaru.
Obecně se hovoří o následujících třídách licencí FOSS:

- Permisivní
- Reciproční (AKA “silný copyleft”)
- Reciproční s omezeným působením (AKA “slabý copyleft”)

Akademické licence

- Relevant subset of popular permissive licenses
- The simplest licenses: very few restrictions
- Reserving only attribution (keep names and copyright notice)
- Available for all uses, including **use in proprietary products**
- Originally written for and popularized by universities

Examples:

- MIT – jQuery
- BSD – FreeBSD, CMake
- ISC – BIND, ISC DHCP

Permisivní licence

- Superset of academic licenses
- Include explicit grant of patent license (in modern variants)
- Available for almost all uses, including use in proprietary products

Examples:

- Apache License – Apache web server, Ant

Reciproční licence

- Requires that derivative work **maintains the same license**
- In most case reciprocal licenses require binary distribution to also include full source code
- Also known as “strong copyleft” or just “copyleft”
- Sometimes called “viral licenses”, as a denigration tactic.
  - If reciprocally licensed code is incorporated, then the application is “infected” and must be released **as a whole** under the same license

Examples:

- GNU GPL – Linux
- AGPL – MongoDB, CiviCRM
- CC BY-SA (for non-software works) – this presentation

Reciproční licence a omezeným působením

- Like reciprocal licenses, but with limits on the scope of which parts of a derived work fall under the license terms
  - changes to the **main work** falls under the license terms
  - **additional works** that happen to be used with/added to/embedded with the main work do not
- They vary in the way the scope of the main work is limited
- According to the denigratory analogy: “virality” is limited to the main work
- Also known as: “weak copyleft”

Examples:

- MPL – Firefox, Libre Office
- CDDL – NetBeans
- LGPL – Qt

What is copyleft?

Copyleft is a strategy of utilizing copyright law to pursue the ***policy goal*** *of fostering and encouraging the equal and inalienable right to* ***copy, share, modify and improve*** *creative works of authorship.*

Copyleft (as a general term) describes any method that utilizes the copyright system to achieve the aforementioned goal. Copyleft as a concept is usually implemented in the details of a specific copyright license, such as the ***GNU General Public License*** *(GPL) and the* ***Creative Commons Attribution Share Alike License****.*

Copyright holders of creative work can unilaterally implement these licenses for their own works to build communities that collaboratively share and improve those copylefted creative works.

— `http://copyleft.org/`

What is copyleft? (cont.)

- Granting the four freedoms is enough to guarantee users will get them **only for a specific copy of the work**
  - how about further downstream redistribution?
  - how about derived works?
  - how about future versions?
- Copyleft makes sure that all users receiving a copy of the program, no matter how modified, also enjoy the four freedoms.
- The copyleft clause might have diverse implementations but all of them (at least for software licenses) share the same concept: **distribution of any version of this program must preserve user freedoms.**
- On the other hand copyleft does preclude **some** business models, and for that reason it gets backlash (e.g., from corporations)

Restrictions and FOSS

Are there permissible restrictions in FOSS licenses?

Restrictions and FOSS

Are there permissible restrictions in FOSS licenses?

Yes: everything that does not get in the way of software freedom is acceptable.
In practice, deciding what is OK and what is not is not always clear cut, and the decisions may very across gatekeepers (FSF/OSI/Debian/etc).
Commonly accepted restrictions are:

- **attribution** of authors (as long as attribution does not impede normal use of the work)
- **transmission of freedoms** (e.g., copyleft)
- detailed protection **of user freedoms** (access to source code or prohibition of “technical measures”, e.g., DRM)

Obsah

1 Spolehlivost/bezpečnost
- Windows
- Qubes OS
- Mikrojádra
  - GNU/Hurd
  - NOVA
- Plan 9

2 Real-time OS

3 Open source
- Ekonomické aspekty
- Základy autorského práva a licence

4 Mobilní OS

Android

Android vs. Linux