Úvod, základní nástroje pro vývoj v prostředí GNU/Linux

Tématicky zaměřený vývoj aplikací v jazyce C
skupina Systémové programování – Linux

Martin Drašar

Ústav výpočetní techniky
Masarykova univerzita
drasar@ics.muni.cz

Brno, září 2015

Představení

RNDr. Martin Drašar, Ph.D.

- výzkumný pracovník Bezpečnostního oddělení Ústavu výpočetní techniky,
- kromě bezpečnosti se věnuji vývoji aplikací pro zpracování vysokorychlostního síťového provozu.

kontakt

email drasar@ics.muni.cz

kancelář FI, budova C, C339 (mezi C315 a C316)

Úvodní informace o kurzu

náplň kurzu a podmínky úspěšného absolvování

Náplň výuky

- obecně vývoj v prostředí GNU/Linux a nástroje, které se vám budou hodit,
- procesy a vlákna,
- komunikace mezi procesy,
- synchronizace a vzájemné vyloučení,
- pokročilé operace se soubory (select, poll, fcntl, ioctl, . . .)
- ladění a debugování aplikací

Cílem není naučit se konstrukce jazyka, které neznáte, ale **vyzkoušet** si vývoj komplexnějších aplikací s využitím toho, co nabízí GNU/Linux.

Administrativní informace

Požadavky na ukončení

- Účastnit se (max. 2 neomluvené absence).
- Odevzdat skupinové úkoly.

Skupinová práce

- Skupiny 2 – 3 lidí: buď se dohodnete, nebo vás rozdělíme.
- Společně pracujete na vývoji kódu a staráte se o jeho úroveň.
- Vytvořte si účet na GitHubu (https://github.com) a vytvořte repozitář `pb173_groupX`, všichni členové skupiny musí mít právo na zápis.

Obecné požadavky na úkoly

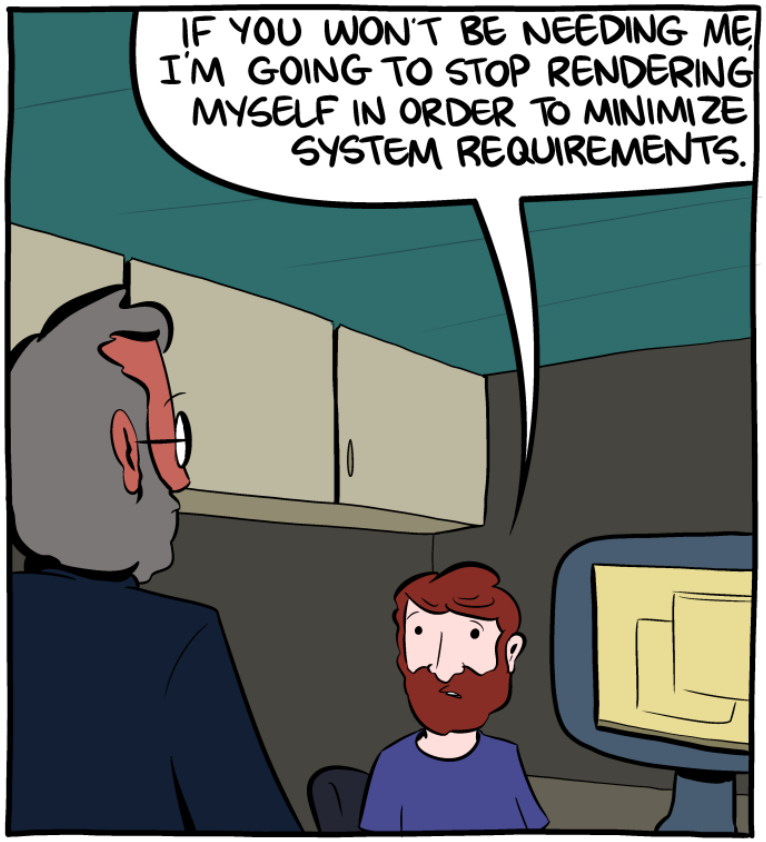
- Úkol odevzdáváte vytvořením tagu s názvem úkolu. Tag nesmí být novější než termín odevzdání.
- Dodržujte coding-style!
- Dokumentujte!
- Neopisujte!

Kultura kódu

coding-style, dokumentace, commit policy

Coding-style I

Myslete na to, že na kódu pracuje většinou několik lidí zároveň.



Fun Fact:
Computer programmers cease to exist
when you're not looking.

Obrázek: www.smbc-comics.com

Coding-style II

Coding style obvykle zahrnuje pravidla pro:

- odsazování
- závorky a mezery
- pojmenování proměnných a funkcí
- komentáře
- používání struktur jazyka – typedef, makra, enum, ...
- návratové hodnoty funkcí
- ...

Coding-style III

Existuje mnoho různých coding-stylů a většinou si dost protiřečí – není důležitý konkrétní formát, ale to, že **je jednotný pro celý kód a že ho dodržují všichni** přispěvatelé projektu.

Skutečné ukázky

- K&R coding style
- Kernel Normal Form (BSD coding style)
  `http://en.wikipedia.org/wiki/Kernel_Normal_Form`
- Linux Kernel Coding Style
  `http://www.kernel.org/doc/Documentation/CodingStyle`
- GNU coding standards
  `http://www.gnu.org/prep/standards/`

Coding-style - ukázka Linux Kernel Coding Style

```
int function(int x)
{
        int a;

        if (x == 0) {
                a = 0;
        }

        switch (x) {
        case '0':
                a = 1;
                break;
        case '1':
                a = 0;
                /* fall through */
        default:
                break;
        }

        return a;
}
```

Dokumentace

Je to opruz, ale **pište komentáře a dokumentaci !!!**

Co je dokumentace:

Dokumentace

Je to opruz, ale **pište komentáře a dokumentaci !!!**

Co je dokumentace:

- komentáře v kódu (doxygen)
- krátká README
- man stránky
- handbooky, tutoriály, howtos – podle zaměření (uživatel/vývojář)

Doxygen

Nástroj pro automatické generování dokumentace.

`www.doxygen.org/`

příklad:

`http://dbus.freedesktop.org/doc/api/html/index.html`

vyzkoušejte si:

`doxywizard(1)`

commit policy

V zásadě jde o pravidla práce s verzovacím systémem (CVS, SVN, GIT, . . .)

- přístupová práva
- pravidla pro adresářovou strukturu
- **pravidla popisu provedených změn**
- `https://techbase.kde.org/Policies/Commit_Policy`

Přenositelnost

aneb kde všude funguje váš kód?

Nikdo nechce, aby váš program fungoval všude, ale musíte vědět **KDE** a **PROČ** nefunguje.

- norma POSIX
- rozšíření kompilátoru
- závislosti na knihovnách/prostředí/architektuře/...

Při řešení problémů s přenositelností dobře slouží buildsystém.

Přenositelnost – datové typy

Problémy

- 32b vs. 64b architektura

  `#include <stdint.h>`

  Normy (C99) udávají pouze minimální velikost jednotlivých datových typů.

  en.wikipedia.org/wiki/C_variable_types_and_declarations#Size

- little vs. big endian (network byte order)

  `#include <endian.h>` (případně `<bits/byteswap.h>`)
  dále funkce jako `ntohs`, `ntohl`, `htons`, `htonl`

Shrnutí a další rady

Modularita kódu

- udržujte pohromadě jen to, co spolu skutečně souvisí
- rozdělujte kód podle funkcionality
- každá funkce by měla dělat jednu konkrétní funkcionalitu
- neduplikujte kód

Dokumentace

- pište komentáře a dokumentaci !!!
- myslete na to, kdo bude dokumentaci číst

Čitelnost

- pozor na magické konstanty – použijte alespoň komentované makro
- pozor na mrtvý kód – debugovací kód, stará funkcionalita

Úvod do prostředí GNU/Linux

buildsystémy, gcc, zdroje informací

Buildsystémy – základní přehled

- Netriviální programy tvoří desítky souborů, závislostí, knihoven, ...
- C/C++ má tragicky pomalý překlad, při změně chcete překládat co nejmíň.
- Potřebujete program, který vám umožní složitost zvládnout.

- make
- GNU Autotools
- CMake
- ninja
- jiné jazyky: ant, gradle, dub, ...

CMake

- Meta-build systém.
- Deklarativní popis procesu.
- Generuje buildovací instrukce pro cílový systém (makefile, VS solution, . . .)
- http://www.cmake.org/

CMake

- Základem je soubor CMakeLists.txt.
- `# cmake . && make`
- Minimální build:

```
cmake_minimum_required (VERSION 2.8)
project ("Test")
add_executable(test test.cpp)
```

GCC

- překladač nejen jazyka C
- standardní překladač na dnešních UNIX-like systémech
- dostupný pro desítky platforem

```
# gcc -o jmeno_programu <vstupni_soubory>
```

GCC - důležité přepínače

Více najdete v man stránce

-std= – použití konkrétního standardu jazyka (gnu89)

-c – nepouštět linker (pouze překlad)

-g – přidat debugovací informace

-pg – přidat profilovací instrukce

-O<num> – optimalizace kódu

-I – cesta kde hledat hlavičkové soubory

-l<lib> – přilinkovat knihovnu

-L<path> – cesta kde hledat knihovny

-D<makro> – definice makra

-Wall – zapne hlavní sadu varování

-WExtra – zapne další důležitá varování

Závěr

Zdroje

Zdroje

obecně

- **používejte man stránky!**
  - 0 – hlavičkové soubory
  - 1 – uživatelské příkazy (aplikace)
  - 2 – systémová volání
  - 3 – knihovní funkce
  - 4 – speciální soubory (`/dev`)
  - 5 – formáty souborů, popis protokolů, ...
  - 6 – hry
  - 7 – různé
  - 8 – systémové příkazy (systémoví daemoni)

Zdroje

obecně

- **používejte man stránky!**
  - 0 – hlavičkové soubory
  - 1 – uživatelské příkazy (aplikace)
  - 2 – systémová volání
  - 3 – knihovní funkce
  - 4 – speciální soubory (`/dev`)
  - 5 – formáty souborů, popis protokolů, ...
  - 6 – hry
  - 7 – různé
  - 8 – systémové příkazy (systémoví daemoni)
- info(1)
- zdrojáky (/usr/src/linux)
- Google

Zdroje

- http://www.stack.nl/~dimitri/doxygen/manual/index.html
- www.gnu.org/software/make/manual/
- http://www.cmake.org/documentation/