Úvod, základní nástroje pro vývoj v prostředí GNU/Linux

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

Petr Velan

Fakulta informatiky
Masarykova univerzita
velan@ics.muni.cz

Brno, září 2014

Představení

RNDr. Petr Velan

- odborný pracovník ve skupině Analýzy provozu sítě na ÚVT
- výzkumný pracovník sdružení CESNET, z.s.p.o,
- dlouhodobě se věnuji vývoji a výzkumu technologií pro monitoring počítačových sítí v prostředí GNU/Linux.

kontakt

email velan@ics.muni.cz
kancelář FI MUNI, místnost C317

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,
- návaznost na PV065 UNIX – programování a správa systému I
- 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 aplikací s využitím toho, co nabízí GNU/Linux.

Administrativní informace

Požadavky na ukončení

- účast je povinná (max. 1 neomluvená absence)
- domácí úkoly a úkoly z hodiny

Odevzdávání úloh – SVN

- fakultní SVN `https://fadmin.fi.muni.cz/auth/sys/svn_ucty.mpl`
- Vytvořte repozitář `pb173` kam budete commitovat svou práci
- Pro tento repozitář přidejte právo číst uživateli `xvelan`
- slidy, různé zdrojáky a další materiály najdete ve studijních materiálech pro tuto skupinu (04)

Obecné požadavky na úkoly

- Odevzdávají se zdrojové soubory, **ne binárky**.
- Spolu se zdrojovými soubory se odevzdává i **Makefile**.
- Dodržujte coding-style!
- Dokumentujte!
- Neopisujte!

Kultura kódu

coding-style, dokumentace, commit policy

Coding-style I

Myslete na to, že kód po vás bude číst někdo další.

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 II

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;
}
```

domácí úkol (do příští hodiny)

- Projděte si různé coding styly a pro jeden se rozhodněte (případně si ho upravte podle svých potřeb).
- Do své SVNky uložte popis svého coding stylu a dodržujte ho v dalších úlohách.

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**
- `http://techbase.kde.org/Policies/SVN_Commit_Policy`

Přenositelnost

aneb kde všude funguje váš kód?

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

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

Při řešení problémů s přenositelností dobře slouží Autotools (viz. první lekce).

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

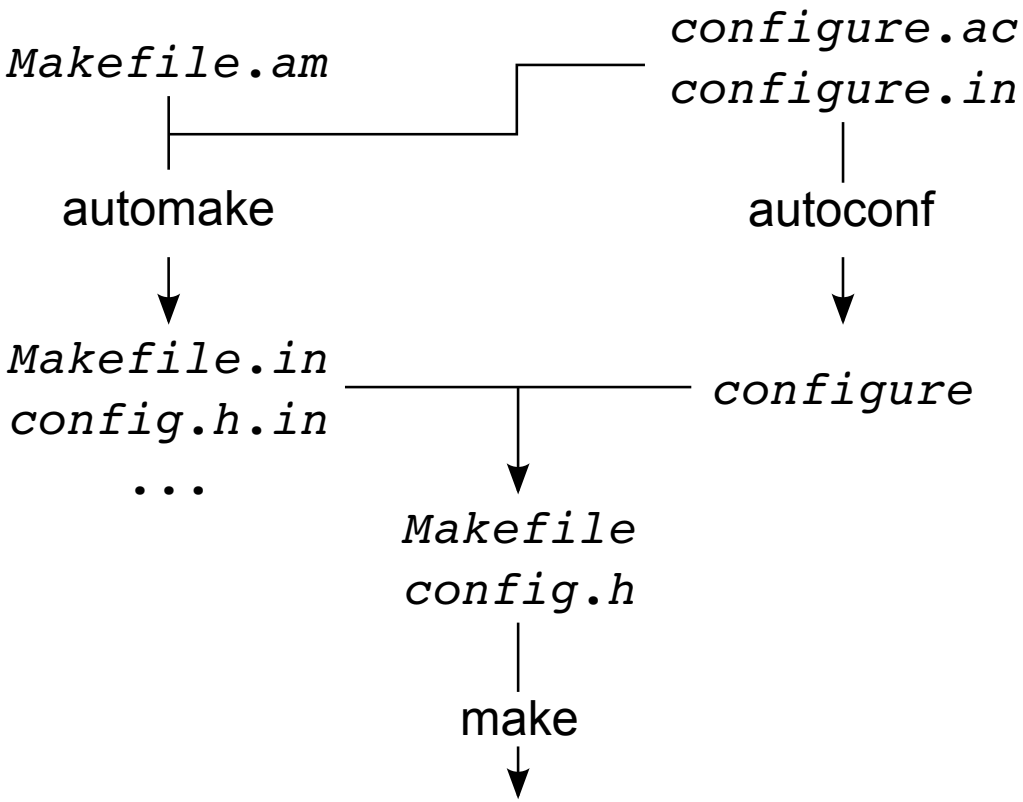
Čitelnost

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

Úvod do prostředí GNU/Linux

autotools, gcc, dynamické knihovny, zdroje informací

GNU Autotools – základní přehled



Cílem je usnadnit práci tvůrci aplikace a uživateli umožnit standardně přeložit a nainstalovat aplikaci:

```
# ./configure && make && make install
```

GNU Autoconf

- makro procesor (převlečený GNU M4)
- kromě nástroje `autoconf` obsahuje balík i další nástroje (`autoreconf`, `autoscan`, `aclocal`, `...`)
- `autoreconf` zaručí spuštění všech potřebných nástrojů GNU Autoconf
- vstupem je soubor `configure.in`
- výstupem je soubor `configure`
- www.gnu.org/software/autoconf/manual/autoconf.html

configure

- shell skript pro kontrolu konfigurace překladového prostředí (programy, knihovny, hlavičkové soubory, ...)

GNU Automake

- generování Makefile.in ze šablony
- obsahuje sadu specifických proměnných pro `make`
- vstupem je `Makefile.am` a `configure.in`
- výstupem je soubor `Makefile.in`
- občas problémy s m4 makry (program `aclocal`)
- poměrně často se tento krok přeskakuje a ručně se upravuje až existující `Makefile.in`

Makefile.am:

```
AUTOMAKE_OPTIONS = foreign
bin_PROGRAMS = mybinary
mybinary_SOURCES = main.c
```

GNU Automake

- generování Makefile.in ze šablony
- obsahuje sadu specifických proměnných pro `make`
- vstupem je `Makefile.am` a `configure.in`
- výstupem je soubor `Makefile.in`
- občas problémy s m4 makry (program `aclocal`)
- poměrně často se tento krok přeskakuje a ručně se upravuje až existující `Makefile.in`

Makefile.am:

```
AUTOMAKE_OPTIONS = foreign
bin_PROGRAMS = mybinary
mybinary_SOURCES = main.c
```

configure.in:

```
AM_INIT_AUTOMAKE(hello, 0.1)
AC_OUTPUT(Makefile)
```

GNU Make (I)

- automatický „překlad“
- nejen překlad, často generování dokumentace a další

soubor Makefile

- proměnné: `CC = gcc` a použití `$(CC)`
- cíle – co se má vytvořit
- závislosti – za jakých podmínek se to má vytvořit
- pravidla/příkazy – jak se to má vytvořit (začínají **Tab**!!!)
- `# make -f Makefile target`
- dále funkce, falešné cíle, podmínky, . . .

běžné cíle

- all
- `%.o: %.c`
- clean

GNU Make (II)

implicitní pravidla

- `$(CC) $(CPPFLAGS) $(CFLAGS) -c`
- `$(CC) $(LDFLAGS) n.o $(LOADLIBES) $(LDLIBS)`
- ...

implicitní proměnné

- `$(CC)`
  – kompilátor jazyka C
- `$(CPP)`
  – C preprocessor
- `$(CFLAGS)`, `$(CPPFLAGS)`
  – flagy/parametry pro kompilátor/preprocesor
- `$(LDFLAGS)`
  – flagy pro kompilátor při linkování
- ...

úkol

- Napište program helloworld
- Připravte si pro něho configure.in a Makefile.am
- Tip: využijte autoscan, autoreconf

úkol

- Napište program helloworld
- Připravte si pro něho configure.in a Makefile.am
- Tip: využijte autoscan, autoreconf
- přidejte do `configure.in` parametr `--enable-debug`, který přidá do vygenerované binárky debugovací informace
- Tip: AC_ARG_ENABLE

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

úkol

- použijte vytvořený helloworld a jeho configure/Makefile
- vyzkoušejte si různé parametry gcc – které parametry mají vliv (a jaký) na velikost výsledného programu?

Závěr

domácí úkoly a zdroje

Domácí úkol

- dokončete co jste nestihli na cvičeních
- projděte si odkazované zdroje pro prácí s autotools a připravte si šablonu vlastního Makefilu (Makefile.*, configure.in) pro úkoly během semestru

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

- www.freshsourcing.sk/content/48
- www.lrde.epita.fr/~adl/dl/autotools.pdf
- www.gnu.org/software/make/manual/
- www.gnu.org/software/automake/manual/
- www.root.cz/serialy/gnu-pomoc-pri-tvorbe-programu/