

# Návaznost jazyka C na OS

Přednášeno v rámci předmětu PB071

Mgr. Šimon Tóth

Fakulta informatiky @ Masarykova Univerzita

3. prosince 2012

# Návaznost jazyka C na OS

- 1 Historické návaznosti
  - Programovací jazyky a jejich vztah k OS
  - Unix a Windows
  - Further reading...
  
- 2 Práce s OS v jazyce C
  - Podpora v jazyce
  - POSIX

# Přístup programovacích jazyků k multi-platformnosti

## Multiplatformní framework

- framework zastřešuje práci s OS
- aplikační kód staví na frameworku
- Jazyky: Java, .Net/Mono

# Přístup programovacích jazyků k multi-platformnosti

## Multiplatformní framework

- framework zastřešuje práci s OS
- aplikační kód staví na frameworku
- Jazyky: Java, .Net/Mono

## Specializované jazyky

- specializované využití
- multiplatformnost buďto nemá smysl uvažovat
- Jazyky: AWK, Javascript

# Přístup programovacích jazyků k multi-platformnosti

## Multiplatformní framework

- framework zastřešuje práci s OS
- aplikační kód staví na frameworku
- Jazyky: Java, .Net/Mono

## Specializované jazyky

- specializované využití
- multiplatformnost buďto nemá smysl uvažovat
- Jazyky: AWK, Javascript

## Skriptovací jazyky

- program se vykonává skrz interpret/virtuální stroj
- multiplatformní v závislosti na interpretu
- vazby na OS jak specifické tak obecné
- Jazyky: Perl, Python, PHP

# Přístup programovacích jazyků k multi-platformnosti

## Multiplatformní framework

- framework zastřešuje práci s OS
- aplikační kód staví na frameworku
- Jazyky: Java, .Net/Mono

## Specializované jazyky

- specializované využití
- multiplatformnost buďto nemá smysl uvažovat
- Jazyky: AWK, Javascript

## Multiplatformní jazyk

- multiplatformnost přímo v návrhu jazyka
- velmi malé množství předpokladů
- vazby na OS přenechány knihovnám
- Jazyky: C, C++

## Skriptovací jazyky

- program se vykonává skrz interpret/virtuální stroj
- multiplatformní v závislosti na interpretu
- vazby na OS jak specifické tak obecné
- Jazyky: Perl, Python, PHP

# Unix vs. Windows

- kořeny Windows začínají u jazyka Basic
- C podporováno v Microsoft C od roku 1983
- rychle přebito jazykem C++

# Unix vs. Windows

- kořeny Windows začínají u jazyka Basic
- C podporováno v Microsoft C od roku 1983
- rychle přebito jazykem C++
  - první podpora C 6.0 (1989)
  - plnohodnotná podpora C/C++ 7.0 (1992)
  - Visual C++ (1993)



# Unix vs. Windows

- kořeny Windows začínají u jazyka Basic
- C podporováno v Microsoft C od roku 1983
- rychle přebito jazykem C++
  - první podpora C 6.0 (1989)
  - plnohodnotná podpora C/C++ 7.0 (1992)
  - Visual C++ (1993)
- pro porovnání
  - ANSI C 1989 / ISO C 1990
  - C99 (1999)
  - ANSI C++ 1998 / revize 2003
  - C++11 (2012)

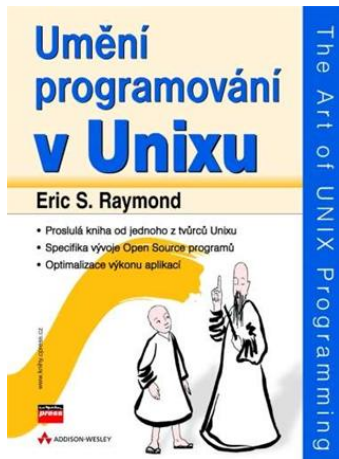
# Aktuální stav ve Windows

- podpora pouze pro ANSI C 89
- pouze Windows API

# Unixový svět

- norma POSIX
  - kterou musí každý certifikovaný UNIX splňovat
  - jazyk C je součástí POSIXu
- historie
  - POSIX.1 (1988)
  - POSIX:2008 (aktuální norma)

# Doporučené čtení



# Práce s OS v jazyce C

- 1 Historické návaznosti
  - Programovací jazyky a jejich vztah k OS
  - Unix a Windows
  - Further reading...
- 2 Práce s OS v jazyce C
  - Podpora v jazyce
  - POSIX

# Manipulace se soubory

- bloková a textová práce se soubory
- manipulace se samotnými soubory `stdio.h`
  - `FILE* tmpfile(void);`
  - `int rename(const char*, const char*);`
  - `int remove(const char*);`

# Práce s časem

## ■ unixový čas `time_t`

- počet vteřin od 1.1. 1970

- `time_t time(time_t *tloc);`

## ■ struktura pro ukládání času `struct tm`

- `time_t` můžeme převést na tuto strukturu

- `struct tm *gmtime(const time_t *clock);`

- `struct tm *localtime(const time_t *clock);`

## ■ textové reprezentace

- `char *ctime(const time_t *clock);`

- `char *asctime(const struct tm *tm);`

# Práce se signály

- `int raise(signal sig);`
- `void (*signal(int sig, void (*func)(int)))(int);1`

---

<sup>1</sup>`demo signals.c`



# Spouštění externích programů

- základní podpora

- synchronní, blokující

- `int system(const char* cmd);`<sup>2</sup>

---

<sup>2</sup>demo ls.c

# Práce s locales

- možnost práce s locales
- samotné locales jsou platformně závislé
- `char *setlocale(int cat, const char* locale);`<sup>3</sup>
- `struct lconv *localeconv(void);`

---

<sup>3</sup>locale.c

# POSIX

- rozšíření standardní knihovny C
  - garance thread safe
  - reentrant verze funkcí
  - nové funkce
- systémové operace
  - API varianty všech cmdline příkazů
  - práce s vlákny
  - práce s procesy
  - komunikace (lokální i síť)
  - a další....

# Rozšíření standardní knihovny

- množství nových funkcí

- `ssize_t getline(char **line, size_t *n, FILE *);`<sup>4</sup>

- `char *strdup(const char *s);`

---

<sup>4</sup>`demo_posix.c`

# Systémové operace

## ■ API varianty unixových příkazů

- `int nice(int incr);`<sup>5</sup>
- `int kill(pid_t pid, int sig);`
- `int chmod(const char *path, mode_t mode);`

## ■ práce s filesystemem

- `dirent.h`<sup>6</sup>
- `sys/stat.h`<sup>7</sup>

---

<sup>5</sup>`demo nice.c`

<sup>6</sup>`demo dirent.c`

<sup>7</sup>`demo stat.c`

# Spouštění programů

## ■ asynchronní, možnost komunikace

- `FILE* popen(const char* cmd, const char* mode);`<sup>89</sup>
- `int pclose(FILE*);`

---

<sup>8</sup>demo calc.c

<sup>9</sup>demo size.c

# Rodina fork/exec

- rodina funkcí pro vytváření a správu procesů<sup>10</sup>
- `pid_t fork(void);`
- `pid_t wait(int *status);`
- `pid_t waitpid(pid_t pid, int *stat, int opt);`
- `int dup(int fildes);`
- `int dup2(int fildes, int fildes2);`
- ...

---

<sup>10</sup>`demo` `watcher.c` `pocet.c` `fork.c`

# Doporučené čtení

- manuálové stránky
  - `man funkce`
  - `man soubor.h`
- Wikipedie



# To je pro dnes vše...

# Mgr. Šimon Tóth

- Gotex (Šumavská 15)
- kanceláře Cesnet/CERIT/UVT (3. poschodí)
- stejný blok jako Sitola a LaBAK

- Ph.D. student / externí lektor
- Fakulta informatiky MU
- toth@fi.muni.cz
- tel. 549 49 6446

- Výzkumný pracovník
- Cesnet z.s.p.o.
- simon@cesnet.cz
- tel. 234 680 235