Operační systémy 1

1. Cvičení

Radek Janoštík

KATEDRA
INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Úvod

- V předmětu se seznámíme s činností procesoru, překladu programů a základního rozhraní operačních systémů
- Budeme aplikovat znalosti z přednášky
- Předpokládá se znalost jazyka C a základní uživatelské práce v operačních systémech Windows i Linux
- Dvě hodiny týdně $\rightarrow$ krátký výklad a zadání samostatné práce
- Dr. Krajča bude zveřejňovat výukový text, který je dobrý si projít předem

Konzultace, kontakt

- Email: radek.janostik@upol.cz
- Pracovna: 5.073
- Telefon: 585 634 711
- Web: `https://apollo.inf.upol.cz/~janostik/`
- Konzultace: Úterý 9:30 – 10:55 nebo dohodou

HW a SW nároky

- Cvičení budou probíhat jak v operačním systému Linux tak Windows
- Assembler s instrukční sadou `AMD64`, některé úkoly nepůjdou na jiných platformách (Apple M1+)
- Na počítačích v učebně oba OS
- Linuxový server `os.inf.upol.cz`
  - Přihlášení pomocí SSH
  - údaje ve tvaru login@PRFAD a heslo stejné jako do domény
- Vhodné mít oba OS (např. pomocí virtualizace)

Podmínky zápočtu

- 2 zápočtové písemky, každá za 10 bodů
  - Jedna opravná, počítají se 2 nejlepší výsledky
  - Z učiva probíraného na cvičeních
- Za aktivní účast na cvičeních možno získat 0.5 bodu
  - = přítomnost na učebně a řešení zadaných úkolů
  - Účast na cvičeních není povinná, ale je silně doporučovaná.
- K udělení zápočtu je nutné získat minimálně 70 % bodů
- Pokud se studentovi či studentce nepodaří získat zápočet pomocí bodů z písemek a cvičení, může mu nebo jí být přidělen bonusový úkol hodnocený až 5 body. Zadání bonusového úkolu je podmíněno získáním alespoň 55 % bodů za cvičení a alespoň 50 % z písemek.

Jazyk C

- Dnes chápán jako nízkoúrovňový jazyk
- Staticky typovaný $\Rightarrow$ každá proměnná má určený datový typ
- Možnost převádět mezi typy (možná ztráta informace)
- Každý datový typ má danou velikost (v bajtech – „kolik zabírá v paměti“)
  - Může se lišit dle platformy a překladače !!!
- Pro práci doporučuji nějaké vývojové prostředí (Eclipse, CodeBlocks, Netbeans, VisualStudio)
  - Já budu používat CLion od Jetbrains

Skalární datové typy

- Pro celá čísla `char`, `short`, `int`, `long`, `long long`
  - navíc neznaménková varianta s klíčovým slovem `unsigned`
  - Co to znamená?
- Rozdílná velikost `long` a `unsigned long` na Windows a Unixech
- Datový typ `char` pro znaky, ale je to obecné 1 bajtové číslo

```
1 char foo = 'A';
2 char foo = 65;
```

- Čísla s plovoucí řádovou čárkou: `float`, `double` a `long double`

Ukazatele

- Hodnota ukazatele ukazuje na místo v paměti, kde je uložena hodnota daného typu

```
1 int a = 42; // proměnná typu int
2 int *p = &a; // ukazatel na hodnotu typu int
3           // operátor & (reference) je použit k získání
                ukazatele (adresy) proměnné a
4 printf("%i\n", *p); // přečtení hodnoty dané ukazatelem p
5                 // operátor * (dereference) získá hodnotu
6 *p = 123; // dereference použita ke změně hodnoty dané
      ukazatelem p
```

- Využití:
  - předávání parametru odkazem
  - práce s poli
  - práce s dynamicky alokovanou pamětí

Předávání argumentů odkazem

- Je vhodné v situacích, kdy potřebujeme uložit výsledek do připravené paměti
- Případně návratu hodnoty přes argument – více návratových hodnot

```
1 void add(int a, int b, int *x) {
2    *x = a + b // změníme hodnotu dané ukazetelm x
3 }
4
5 int z;
6 add(10, 20, &z); // předáme adresu z
```

Pole

- Pole = celými čísly indexovaná kolekce hodnot stejného typu
- Pevná nebo nespecifikovaná velikost

```
1 int a[3]; // pole celých čísel o třech prvcích
2 int b[]; // pole celých čísel s nespecifikovanou velikostí
```

- Pole nemá údaje o své velikosti
- Jazyk nekontroluje, zda nepřistupujeme „za pole“
- Následující se sice provede, ale nespecifikované chování

```
1 int a[3];
2 a[10] = 42;
```

Pole a ukazatele

- Pole můžeme chápat jako ukazatel na první prvek pole.
- Aritmetika s ukazateli nám pak umožňuje pole procházet

```
1 int a[3];
2 a[0]= 1; // priřadí prvnímu prvku pole hodnotu 1
3 *a = 1; // to samé s využitím ukazatele
4 a[1] = 2; // priřadí druhému prvku pole hodnotu 2
5 *(a + 1) = 2; // to samé s využitím ukazatele
6 int *b = a; // přiřadí do b začátek pole a
7 b++; // pole b bude začínat na druhém prvku pole a
8 printf("%i\n", b[0]); // vypíše 2
```

Dynamická alokace paměti

- funkce `void *malloc(size_t size)` – alokuje minimálně size bajtů paměti
  - Vrací ukazatel na alokovanou paměť

```
1 int *p = (int *) malloc(sizeof(int));
2      // alokuje pamět pro jednu hodnotu typu int
3      int *a = (int *) malloc(sizeof(int) * 10);
4      // alokuje pole typu int o velikosti deset prvků
```

- Alokovanou paměť musíme uvolnit pomocí funkce `free`
- Alokovaná paměť není inicializovaná, může obsahovat „cokoliv“
- `calloc`, `realloc`

Řetězce

- Nejsou v jazyce C základním typem, ale pouze jako pole znaků
- Tedy ukazatele typu `char *` – ukazuje na první znak řetězce
- Konec řetězce indikován znakem `'\0'` (hodnota 0)
- U řetězcových literálů je konec řetězce doplněn automaticky
  - Často mohou být neměnné
  - Nelze provést:

```
1        char *s = "abc";
2        s[0] = 'A'; // pravděpodobně selže
```

Pravdivostní hodnoty

- Není vyhrazen *speciální* datový typ
- Možno použít všechny ostatní datové typy
- `0` nebo `NULL` chápány jako *nepravda*
- Jiné hodnoty chápány jako *pravda*

```
1 int a = 1;
2 int *p = NULL;
3 if (a == 2) { } // explicitní porovnání
4 if (a) { } // test, jestli proměnná ,,a'' obsahuje nenulovou
      hodnotu
5 if (!p) { } // test, zda je ukazatel roven NULL
```

Strukturované datové typy

- Spojení souvisejících hodnot do jednoho datového typu
- Vytvoření:

```
1 struct point {
2     int x;
3     int y;
4 };
```

- Použití:

```
1 struct point a = { 2, 4 };
2 void point_print(struct point p) {
3     printf("[%i, %i]\n", p.x, p.y);
4 }
```

- Do funkcí se předávají hodnotou (kopie) nikoliv odkazem
- Alias pomoci `typedef`

Operátory

- Nástroje jazyka pro manipulaci s datovými typy (operace – sčítání, odčítání, násobení, . . .)
- `a = b` (přiřazení) přiřadí do prvního operandu hodnotu druhého operandu a vyhodnotí se na jeho hodnotu
- Rozdíl mezi inkrementací:`a++` a `++a`. Analogicky dekrementace
- `a ? b : c` (podmíněný výraz, ternární operátor) – pokud je první operand pravda, vyhodnotí se na druhý operand, jinak se vyhodnotí na třetí operand
- `a && b` (logický součin) – vyhodnotí se na pravda, pokud jsou oba operandy pravdivé, jinak se vyhodnotí na nepravda
- `a || b` (logický součet) – vyhodnotí se na pravda, pokud je alespoň jeden operand pravda, jinak se vyhodnotí na nepravda
  - Zkrácené vyhodnocení

Bitové operace

- Práce s jednotlivými bity (nejnižší úroveň)
- `&` (bitový součin), `\` — (bitový součet), `^` (bitová non-ekvivalence, výlučné nebo, XOR), `\~` (negace,inverze bitů), `<<` a `>>` (bitové posuny).

```
  0101 0011          0101 0011          0101 0011
& 1001 0010        | 1001 0010        ^ 1001 0010
-----------        -----------        -----------
  0001 0010          1101 0011          1100 0001
```

```
  0101 0011          1001 0010          1001 0010
 <<       1        >>        1        >>        1
------------       ------------       ------------
  1010 0110          1100 1001          0100 1001
                    znaménková        neznaménková
                     varianta           varianta
```

Úkol (1/2)

1. Napište funkci `void int2bits(char *, int)`, která převede číslo na textový řetězec představující jeho zápis v binární podobě.
2. Napište funkci `int bits2int(char *)`, která převede textový řetězec představující zápis čísla v binární podobě (tj. 010110010010...) na hodnotu typu `int`.
3. Implementujte funkci `void my_memcpy(void *dest, void *src, size_t size)`, která se chová jako funkce `memcpy` a přenese po jednotlivých bajtech obsah paměti z jednoho místa na druhé, předpokládejte, že úseky paměti se nepřekrývají.

Úkol (2/2)

1. Navrhněte vhodnou strukturu pro spojový seznam obsahující dvě hodnoty jméno (textový řetězec) a věk (celé číslo). Napište funkci, která bude přidávat prvky do seznamu a funkci, která vypíše obsah tohoto seznamu.
2. Napište funkci `short encode_date(char day, char month, short year)`, která zakóduje datum do 16bitového čísla následovně: `YYYY-YYYM-MMMD-DDDD`.
3. Napište funkci `void decode_date(short date, int *day, int *month, int *year)`, která dekóduje datum vytvořené předchozí funkcí a vrátí hodnoty pomocí předaných ukazatelů.
4. Napište funkci, která zjistí, v jakém pořadí jsou vyhodnocovány argumenty.