Zápočtové úkoly

6. prosince 2022

1 Roura pro vlákna

Naprogramujete objekt, který bude fungovat jako roura, kterou si mohou vlákna předávat data. Tento objekt bude reprezentován typem `struct pipe` a bude se s ním pracovat pomocí sady pěti funkcí.

Funkce `struct pipe *pipe_create(unsigned int size)` vytvoří objekt roury, inicializujej a vytvoří bufferu o velikosti `size` bytů, kterým se budou předávat hodnoty z jednoho konce roury do druhého.

Funkce `unsigned int pipe_write(struct pipe *pipe, unsigned char *data, unsigned int size)` zapíše do roury `size` bytů předaných argumentem `data`. Pokud při zápisu dojde k zaplnění bufferu roury, dojde k uspání vlákna v dané funkci do doby, než se v bufferu uvolní místo. Funkce vrací počet bytů, které byly skutečně zapsány do roury.

Funkce `unsigned int pipe_read(struct pipe *pipe, unsinged char *data, unsigned int size)` přečte z roury `pipe` nejvýše `size` bytů a uloží je do paměti dané ukazatelem `data`. Pokud není v bufferu roury dostatek dat, které by mohly být přečteny, je vlákna uspáno a probuzeno, když se bufferu data v objeví. Funkce vrací počet skutečně přečtených bytů.

Funkce `void pipe_close(struct pipe *pipe)` uzavře rouru pro čtení i zápis. Po zavolání této funkce není možné do roury zapisovat a funkce `pipe_write` vrátí 0. Funkce `pipe_read` vrátí data uložená v bufferu. Pokud dojde k zavolání `pipe_close` a některá z funkcí `pipe_read` nebo `pipe_write` čeká než bude moci zapsat do bufferu nebo z něj číst, je toto čekání přerušeno a funkce vrací skutečný počet zapsaných nebo přečtench bytů.

Funkce `void pipe_free(struct pipe *pipe)` uvolní všechny prostředky spojené s rourou `pipe`.

Poznámky k řešení:

- Funkce `pipe_write` a `pipe_read` by měly používat pasivní čekání. Je možné použít i aktivní čekání, bude to však hodnoceno nižším počtem bodů.
- Pokud bude do roury zapisovat více vláken současně nebo z ní více vláken současně číst, je výsledek nedefinován.
- Úlohu vypracujte pro vámi zvolený operační systém.
- Je možné získat až 2 bonusové body, vypracujete-li úlohu pro Windows i Linux.

- Je možné získat až 3 bonusové body, vypracujete-li úlohu jako multiplatformní[1] pro Window a Linux.

2 pgmtoascii

Mezi nejjednodušší formáty pro uložení obrazových dat patří formát `PGM`, které slouží k uložení obrázků ve stupních šedé. Existují dvě varianty formátu (i) textový (ASCII) a (ii) binární (RAW). My budeme pracovat s textovou variantou formátu, která ma tvar:[2]

1. Magická hodnota identifikující formát, řetezec "P2",
2. bílé znaky (mezery, `\r`, `\n`, `\t`),
3. šířka obrázku (ASCII řetězec, číslo v desítkové soustavě),
4. bílé znaky,
5. výška obrázku (ASCII řetězec, číslo v desítkové soustavě),
6. maximální stupeň šedé (včetně, ASCII řetězec v desítkové soustavě),
7. bílé znaky,
8. hodnoty jednotlivých pixelů (zleva doprava, zhora dolů, jako ASCII řetězec, číslo v desítkové soustavě).
9. Součástí souboru mohou být i komentáře, cokoliv od znaku # do konce řádku je ignorováno.

Jak vypadá obrázek ve formátu PGM (ASCII) ukazuje Obrázek 1.

Vytvořte program, který bude mít tři argumenty:

1. vstupní obrázek ve formátu PGM
2. výstupní textový soubor
3. paletu, tj. posloupnost znaků, která obsahuje minimálně tolik znaků, kolik má vstupní obrázek stupňů šedé.

Tento program po svém spuštění načte vstupní obrázek do paměti a uloží jej do textové reprezentace tak, že každý řádek souboru bude odpovídat jednomu řádku obrázku a každý znak bude odpovídat jednomu pixelu obrázku. A to tak, že pro každý stupeň šedé se použije jeden znak ze zadané palety.

Ukázka použití:

```
./pgmtoascii pict.pgm pict.txt " .-+=o*0#@"
```

[1] Jeden zdrojový kód, který je možno bez jakýchkoliv změn přeložit a spustit na více platformách.

[2] `https://netpbm.sourceforge.net/doc/pgm.html#plainpgm`

Obsah výstupního souboru můžete vidět jako Obrázek 2.

Poznámky k řešení:

- Pro čtení a zápis do souboru je nutné použít mapování souboru do paměti.
- Je možné získat až 2 bonusové body, vypracujete-li úlohu pro Windows i Linux.
- Je možné získat až 3 bonusové body, vypracujete-li úlohu jako multiplatformní.
- Běžné grafické nástroje umí většinou vytvořit jen obrázky ve 256 stupních šedi. Pokud byste si chtěli vytvořit vlastní testovací obrázky, můžete použít sadu nástrojů *netpbm*, která je k dispozici v unixových operačních systémech.

  ```
  pamdepth pocet-stupnu-sede obrazek.pgm  | pnmtopnm -plain > upraveny-obrazek.pgm
  pamdepth 10 foo.pgm  | pnmtopnm -plain > bar.pgm
  ```

- Pokud je na vstupu nevalidní obrázek, je chování nedefinováno. Pokud je na vstupu příliš malá paleta, je o tom uživatel informován.

3 Alokátor pro Lisp

V programovacích jazycích jako je Lisp, Scheme nebo Racket hraje důležitou roli struktura označovaná jako *tečkový pár*[3]. Hodnota typu tečkový pár obsahuje vždy právě dvě hodnoty libovolného typu, tj. může obsahovat čísla, řetězce, další tečkové páry atd. V jazyce C se tečkovým párům nejvíce blíží struktura typu:[4]

```
struct pair {
    void *ar;
    void *dr;
};
```

Pokud budeme používat standardní funkci `malloc` pro alokaci jednotlivých tečkových párů, bude docházet k výraznému plýtvání místem, protože pro každý pár potřebujeme ještě místo pro hlavičku objektu, tj. obvykle o velikosti dalších dvou slov. Úkolem je navrhnout alokátor, který umožní efektivně (s minimální režií) alokovat tečkové páry.

Naprogramujte funkci `struct pair *lalloc()`, která při svém prvním zavolání získá oblast paměti vhodné velikosti, a vrací odkazy na volné úseky paměti typu `struct pair`. Pokud není možné další úseky paměti alokovat, vrací `NULL`.

Naprogramujte funkci `void lfree(struct pair *p)`, která uvolní zadaný tečkový pár pro další použití.

Poznámky k řešení:

- Pro získání volného místa, které budete přidělovat, použijte mapovaní (anonymní oblasti) do paměti, použití funkce `malloc` je taktéž možné, avšak bude hodnoceno nižším počtem bodů.

[3] anglicky *dotted pair*

[4] Zachováváme pojmenování, jak se používá v jazycích odvozených od Lispu. Význam jednotlivých atributů nebude pro tento úkol relevantní.

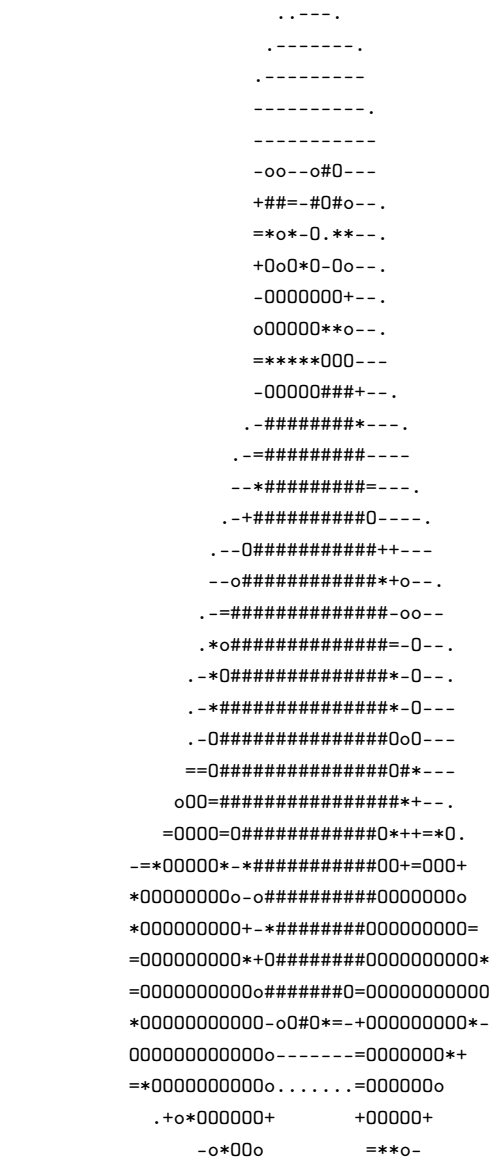
- Pro evidenci volných tečkových párů použijte volné tečkové páry.
- Úlohu vyřešte pro Windows nebo Linux.
- Je možné získat 1 bonusový bod, pokud bude alokátor naprogramován tak, aby jej bylo možné používat z více vláken.
- Je možné získat 1 bonusový bod, pokud bude funkce `lalloc` schopna získávat další oblasti paměti v případě, že paměť získaná při prvním zavolání nebude dostačovat.

Poznámky vztahující se k řešení všech úkolů

- Úlohy odevzdávejte do 15. prosince 2022, 23:59 CET jako email na adresu `petr.krajca (at) upol.cz` s předmětem `OS2:ZAPOCET`.
- Řešení každé úlohy uložte do souboru pojmenovaného `prijmeni-jmeno-cislo_ulohy-(win|lin|any).c`, kde `win` identifikuje řešení pro Windows, `lin` pro Linux a `any` označuje multiplatformní řešení.
- Řešení posílejte jako přílohu emailu. Pokud budete chtít použít komprimační program, použijte buď formát zip nebo kombinaci tar+gzip.
- Každá úloha je hodnocena 5 body.
- Nad rámec těchto bodů je možné získat bonusové body, viz zadání jednotlivých úloh.
- Jako multiplatformní řešení jsou brána jenom ta, kde je společný kód pro obě platformy a platformově závislé věci jsou vhodným způsobem izolovány.
- Při řešení nepoužívejte knihovny, které jdou nad rámec standardní knihovny jazyka C nebo funkcí poskytovaných operačním systémem.
- Kód by měl být srozumitelný a vhodně okomentovaný.
- Všechna řešení projdou kontrolou detekující plagiáty.
- Informace o udělení či neudělení zápočtu vám bude doručena jako odpověď na email se zápočtovými úlohami.

```
P2
32 37
9
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 2 2 2 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 2 2 2 2 2 2 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 2 5 5 2 2 5 8 7 2 2 2 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 3 8 8 4 2 8 7 8 5 2 2 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 4 6 5 6 2 7 1 6 6 2 2 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 3 7 5 7 6 7 2 7 5 2 2 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 2 7 7 7 7 7 7 7 3 2 2 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 5 7 7 7 7 7 6 6 5 2 2 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 4 6 6 6 6 6 7 7 7 2 2 2 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 2 7 7 7 7 7 8 8 8 3 2 2 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 2 8 8 8 8 8 8 8 8 6 2 2 2 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 2 4 8 8 8 8 8 8 8 8 8 2 2 2 2 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 2 2 6 8 8 8 8 8 8 8 8 8 4 2 2 2 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 2 3 8 8 8 8 8 8 8 8 8 8 7 2 2 2 2 1 0 0 0 0 0
0 0 0 0 0 0 0 1 2 2 7 8 8 8 8 8 8 8 8 8 8 3 3 2 2 2 0 0 0 0 0 0
0 0 0 0 0 0 0 2 2 5 8 8 8 8 8 8 8 8 8 8 8 6 3 5 2 2 1 0 0 0 0 0
0 0 0 0 0 0 1 2 4 8 8 8 8 8 8 8 8 8 8 8 8 8 2 5 5 2 2 0 0 0 0 0
0 0 0 0 0 0 1 6 5 8 8 8 8 8 8 8 8 8 8 8 8 8 4 2 7 2 2 1 0 0 0 0
0 0 0 0 0 1 2 6 7 8 8 8 8 8 8 8 8 8 8 8 8 8 6 2 7 2 2 1 0 0 0 0
0 0 0 0 0 1 2 6 8 8 8 8 8 8 8 8 8 8 8 8 8 8 6 2 7 2 2 2 0 0 0 0
0 0 0 0 0 1 2 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 7 5 7 2 2 2 0 0 0 0
0 0 0 0 0 4 4 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 7 8 6 2 2 2 0 0 0 0
0 0 0 0 5 7 7 4 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 6 3 2 2 1 0 0 0 0
0 0 0 4 7 7 7 7 4 7 8 8 8 8 8 8 8 8 8 8 8 8 7 6 3 3 4 6 7 1 0 0
2 4 6 7 7 7 7 7 6 2 6 8 8 8 8 8 8 8 8 8 8 8 7 7 3 4 7 7 7 3 0 0
6 7 7 7 7 7 7 7 7 5 2 5 8 8 8 8 8 8 8 8 8 8 7 7 7 7 7 7 7 5 0 0
6 7 7 7 7 7 7 7 7 7 7 3 2 6 8 8 8 8 8 8 8 8 7 7 7 7 7 7 7 7 4 0
4 7 7 7 7 7 7 7 7 7 7 6 3 7 8 8 8 8 8 8 8 8 7 7 7 7 7 7 7 7 7 6
4 7 7 7 7 7 7 7 7 7 7 7 5 8 8 8 8 8 8 8 8 7 4 7 7 7 7 7 7 7 7 7
6 7 7 7 7 7 7 7 7 7 7 7 2 5 7 8 7 6 4 2 3 7 7 7 7 7 7 7 7 6 2
7 7 7 7 7 7 7 7 7 7 7 7 7 5 2 2 2 2 2 2 2 4 7 7 7 7 7 7 7 6 3 0 0
4 6 7 7 7 7 7 7 7 7 7 7 7 5 1 1 1 1 1 1 1 4 7 7 7 7 7 7 5 0 0 0 0
0 0 1 3 5 6 7 7 7 7 7 7 7 3 0 0 0 0 0 0 0 3 7 7 7 7 7 3 0 0 0 0 0
0 0 0 0 0 0 0 2 5 6 7 7 5 0 0 0 0 0 0 0 0 0 0 4 6 6 5 2 0 0 0 0 0 0
```

Obrázek 1: Obrázek o rozměrech 32x37 pixelů v deseti odstínech šedi ve formátu PGM



Obrázek 2: Ukázkový výstup programu `pgmtoascii`