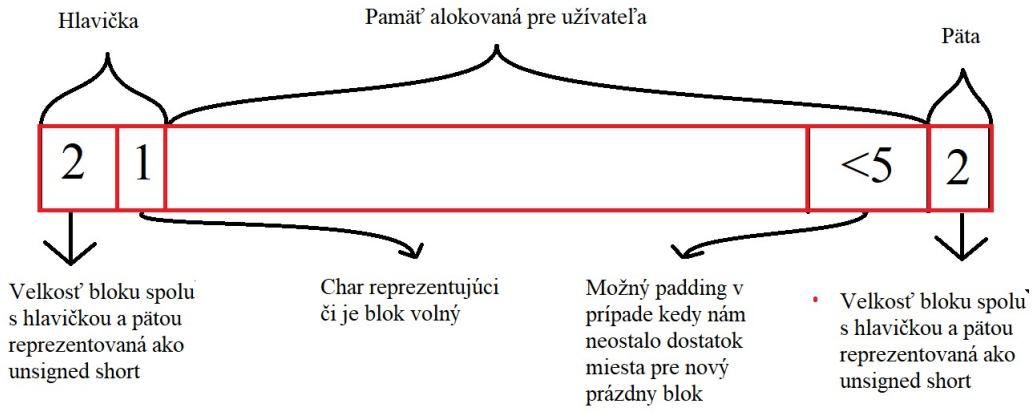
DSA-Zadanie 1 Správca pamäti

Peter Kúdela

15. marca 2020

Metóda riešenia

Implementácia je tvorená štýlom implicitných zoznamov, bloky sú štrukturované nasledovne. (Hodnoty vyjadrené v bajtoch)



Memory_init

Funkcia nastavý hodnotu prvých dvoch bajtov našej pamäťe na jej celú velkosť, z ostatku vytvorý prvý prázdny blok.

Memory_alloc

Po blokoch sa pohybujeme pomocou charového pointra reprezentujúceho súčastnú polohu s názvom curr. Pointer sa vždy nachádza na začiatku hlavičky daného bloku. Všetky bloky postupne prechádzame kým nenájdeme voľný blok dosť veľký alebo vätší pre požadovanú pamäť. Pointer curr posúvame pointrovou aritmetikou. Ak sa v pamäti vhodný blok nenachádza funkcia vráti NULL.

Memory_free

Začneme označením bloku ako voľný. Ďalej sledujeme, či je možné zlučiť náš nový prázdny blok s predchádzajúcim (ak sa nachádzame v prvom bloku tento krok preskočíme) alebo s nasledujúcim (ak sa nachádzame v poslednom bloku tento krok preskočíme) blokom.

Memory_check

Ako prvé v tejto funkcií testujeme, či je block nenulový a či sa nachádza v našom bloku pamäte. Ďalej postupne prechádzame všetky bloky a hladáme zhodu ukazovateľov, ak ju nájdeme vrátime 1, ak nie vrátime 0.

Testovanie

Prideľovanie rovnakých blokov malej veľkosti

Prebehlo podľa očakávaní. V poli pointrov bolo alokovaných prvých x pointrov, ostatné pokusy o alokáciu vrátili null.

Prideľovanie nerovnakých blokov malej veľkosti

Keďže sme tentoraz alokovaly náhodné velkosti, posledný alokovaný pointer sa často nachádzal až po pár pokusoch ktoré vrátili null (požadovali väčšiu pamäť ako bolo možné). Po manuálnom prezretí pramäte pri debugovaní usudzujem že funkcia funguje ako má.

---

prideľovanie nerovnakých blokov väčšej veľkosti

Test prebehol v poriadku.

---

prideľovanie nerovnakých blokov malých a veľkých veľkostí

Taktiež prebehol v poriadku.

```
Test sameSmall
1 1 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 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 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 0 0 0 0 0 0 0 0 0 0
---------------------------------------------
Test randSmall
1 1 1 1 1 1 0 0 0 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 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 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 0 0 0
---------------------------------------------
Test randBig
1 1 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 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 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 0 0 0 0 0 0 0 0 0 0
---------------------------------------------
Test randMix
1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0
---------------------------------------------
```

Výpisy kontrolované pomocou funkcie memmory check (prvý riadok alokácia, druhý uvoľňovanie)

Testovanie Free

Na nasledovných screenshotoch je zobrazený blok pamäte velkosti 32 na ktorých testujeme free, prvé dva bajty na screenshotoch neni vidno.

Prvý blok

Pred

```
0a 00 01 cc cc cc cc cc 0a 00 0a 00 01 cc cc cc cc cc 0a 00 0a 00 01 cc cc cc cc cc 0a 00
```

Po

```
0a 00 00 cc cc cc cc cc 0a 00 0a 00 01 cc cc cc cc cc 0a 00 0a 00 01 cc cc cc cc cc 0a 00
```

Posledný blok

Pred

```
0a 00 01 cc cc cc cc cc 0a 00 0a 00 01 cc cc cc cc cc 0a 00 0a 00 01 cc cc cc cc cc 0a 00
```

Po

```
0a 00 01 cc cc cc cc cc 0a 00 0a 00 01 cc cc cc cc cc 0a 00 0a 00 00 cc cc cc cc cc 0a 00
```

Zľučovanie blokov

Otestované boli aj samostatne zľava aj s prava, nasledujúce screenshoty sú zo zľučovania oboch strán naraz:

Pred

Po

Zložitosť

Priestorová

Každý blok potrebuje 5 bajtov + velkosť alokovaného bloku (3 hlavička, 2 päta).

Časová

V najhoršom prípade O(n) kde n je počet všetkých blokov.