Martin Hric
ID:111696

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

DÁTOVÉ ŠTRUKTÚRY A ALGORITMY
SPRÁVCA PAMÄTE

Vypracoval: Martin Hric

2020/2021

Martin Hric
ID:111696

ÚVOD

V tomto zadaní sme mali za úlohu implementovať v programovacom jazyku C 4 funkcie:

- void *memory_alloc(unsigned int size);
- int memory_free(void *valid_ptr);
- int memory_check(void *ptr);
- void memory_init(void *ptr, unsigned int size);

Ja som si vybral metódu implicitného listu, first-fit. V tejto metóde sa pohybujem cez celý list a skáčem po hlavičkách ako voľných tak aj obsadených blokov pamäti.

First fit znamená, že prvý vhodný blok alokujem, ktorý ak je väčší ako potrebujem tak ho rozdelím.

Ja som si neimplementoval žiadne pomocné funkcie, nakoľko mi to prišlo trochu zbytočné keďže to nemám nejak komplikovane dlhé.

Nepoužívam globálnu hlavičku, koniec regiónu mám označený písmenom ‘k’.

Celú kontrolu som robil v Clione v debuggeri, takže nijak si to nevypisujem cez konzolu.

Martin Hric
ID:111696

memory_init:

Znenie zadania:

Funkcia memory_init slúži na inicializáciu spravovanej voľnej pamäte. Predpokladajte, žefunkcia sa volá práve raz pred všetkými inými volaniami memory_alloc, memory_free amemory_check. Viď testovanie nižšie. Ako vstupný parameter funkcie príde blok pamäte,ktorú môžete použiť pre organizovanie a aj pridelenie voľnej pamäte. Vaše funkcie nemôžupoužívať globálne premenné okrem jednej globálnej premennej na zapamätanie ukazovateľanapamäť, ktorá vstupuje do funkcie memory_init. Ukazovatele, ktoré prideľuje vaša funkcia memory_alloc musia byť výhradne z bloku pamäte, ktorá bola pridelená funkciimemory_init.

V tejto funkcií na začiatku zaokrúhľujem veľkost bloku pamäte na 4 nadol, pretože používam hlavičky a pätičky o veľkosti jeden int, čo má 4 byty, takže sa mi lepšie bude s tým pracovať.

Celý region si nastavím na 0 , aby som to mal prehľadnejšie.

Koniec regiónu označím písmenom ‘k’ aby som nevyletel z regiónu.

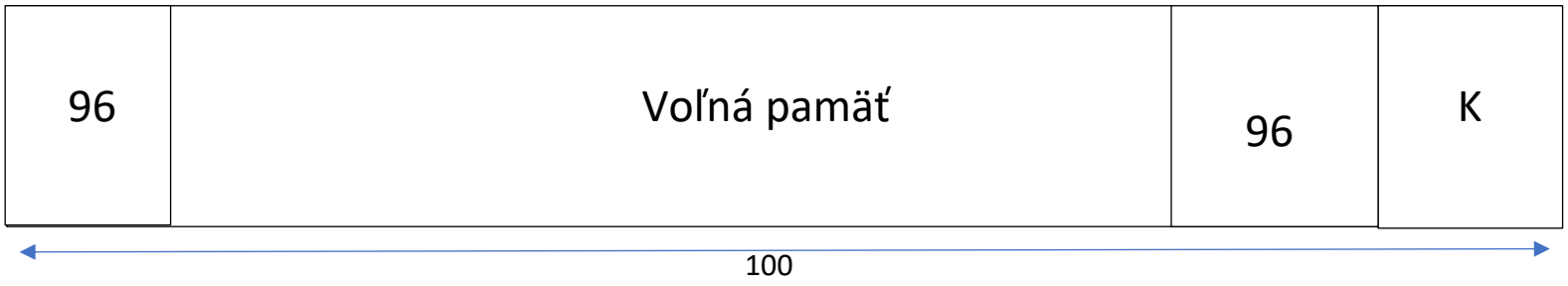
```
void memory_init(void *ptr, unsigned int size){

    if(size%4==1) size=size+3;
    if(size%4==2) size=size+2;            //zaokruhlujem to na 4
    if(size%4==3) size=size+1;

    for(int i = 0 ;i<size;i++){
        *((char*)ptr+i)=0;}                //nastavim cely region na 0
    memory = ptr;

    *(int*)ptr=size-sizeof(int);                    //zaciatok bloku
    *(int*)(ptr+size -2*sizeof(int))=size-sizeof(int);   //koniec bloku
    *(int*)(ptr+size -sizeof(int))='k';                  //koniec regionu
}
```

Martin Hric
ID:111696

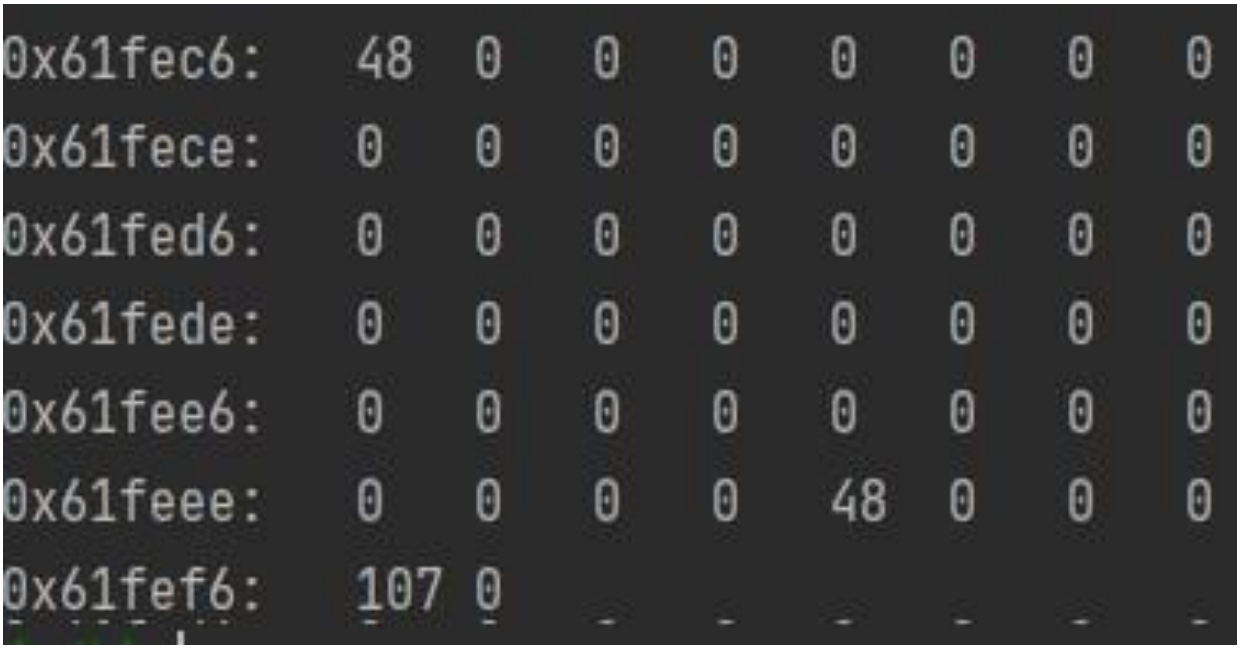
memory_alloc:

Znenie zadania:

Funkcia memory_alloc má poskytovať služby analogické štandardnému malloc. Teda,vstupné parametre sú veľkosť požadovaného súvislého bloku pamäte a funkcia mu vráti:ukazovateľ na úspešne alokovaný kus voľnej pamäte, ktorý sa vyhradil, alebo NULL, keď nieje možné súvislú pamäť požadovanej veľkosť vyhradiť.

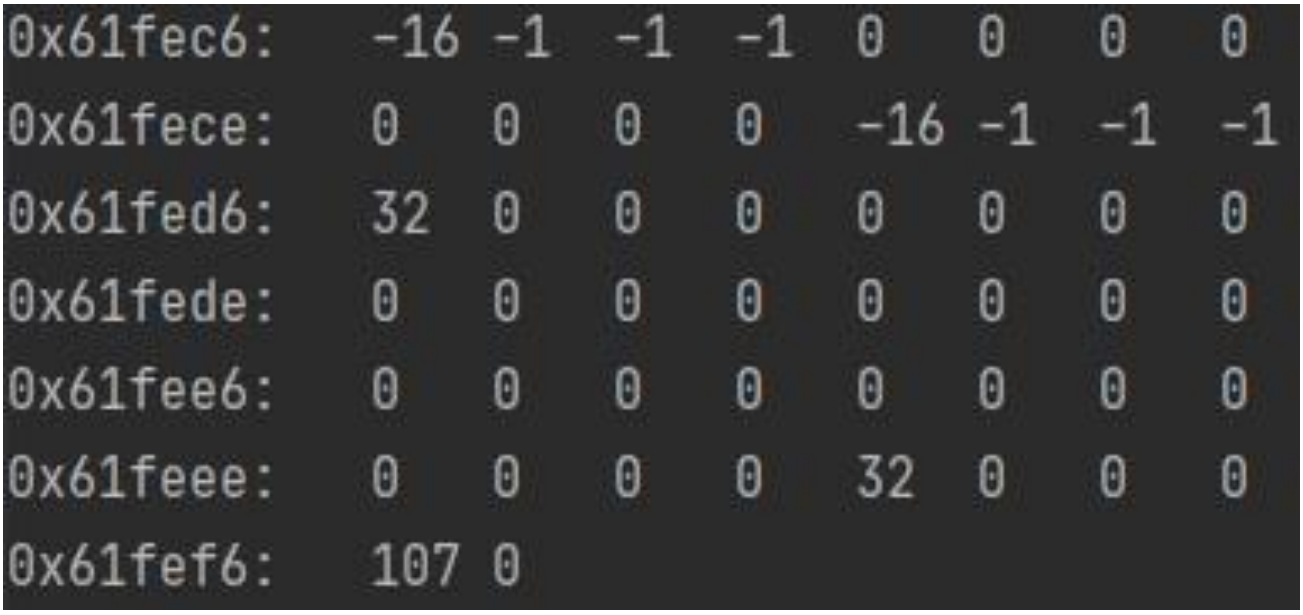


Po funkcii memory_init(region,50);



Martin Hric
ID:111696

po funkcii memory_alloc(7);



Ak globálny pointer je NULL alebo veľkosť, ktorú chcem alokovat je 0, vraciam NULL.

Parameter size zaokrúhľujem na 4.

Zbytok funkcie prebieha jednoducho, while cyklus pokiaľ pomocný pointer nie je na ‘k’, ďalší while cyklus, cez ktorý sa posúvam pokým je hlavička záporná, alebo menšia ako potrebujem.

Ak nájde vyhovujúci blok, nastaví hlavičku, patičku a zapíše zaň veľkosť aká ostane z toho bloku.

Martin Hric
ID:111696

memory_check:

Znenie zadania:

Funkcia memory_check slúži na skontrolovanie, či parameter (ukazovateľ) je platný ukazovateľ,ktorý bol v nejakom z predchádzajúcich volaní vrátenýfunkciou memory_alloc a zatiaľnebol uvoľnený funkciou memory_free. Funkcia vráti 0, ak je ukazovateľneplatný, inak vráti 1.

```
int memory_check(void *ptr){
    if(*(int*)ptr<-3 && *(int*)(ptr+sizeof(int))==0) return 1;
    else return 0;
}
```

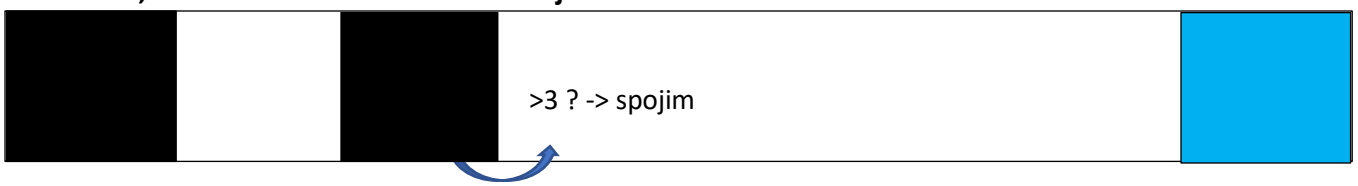
Memory check mám spravený veľmi jednoducho, ak je pointer záporné číslo a zároveň int za pointerom je rovný nule, čiže to nie je patička , tak mi vráti 1, inak 0.

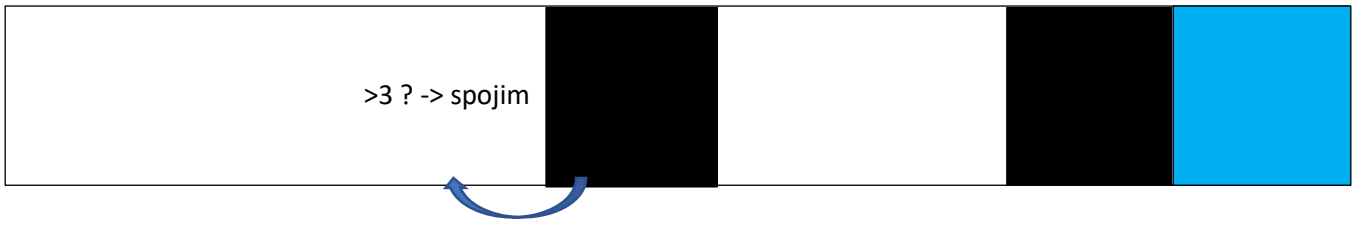
memory_free:

Znenie zadania:

Funkcia memory_free slúži na uvoľnenie vyhradeného bloku pamäti, podobne ako funkcia free. Funkcia vráti 0, ak sa podarilo (funkcia zbehla úspešne) uvoľniť blok pamäti, inak vráti1. Môžete predpokladať, že parameter bude vždy platný ukazovateľ, vrátený z predchádzajúcich volanífunkciememory_alloc, ktorý ešte nebol uvoľnený.

3 situácie, ktoré môžu nastať keď uvoľňujem:





Martin Hric
ID:111696

Testovania:

TEST 1:

Pamäť 50 , rovnaké bloky(8) a ich uvoľnenie, potom alokovanie 24,8,8 , uvoľnenie 24 a alokovanie opäť 8.

V tomto teste mám navyše alokovania, ale to kvôli zisteniu, či naozaj mi alloc vráti NULL.

Tento test mi jediný odhalil chybu. Ako vyzerá pamäť po zbehnutí testu:

```
0x61fec6:   -16 -1  -1  -1  0   0   0   0
0x61fece:   0   0   0   0   -16 -1  -1  -1
0x61fed6:   -16 -1  -1  -1  0   0   0   0
0x61fede:   0   0   0   0   -16 -1  -1  -1
0x61fee6:   0   0   0   0   0   0   0   0
0x61feee:   0   0   0   0   -16 -1  -1  -1
0x61fef6:   107 0
```

TEST 2:

Pamäť 500, náhodné bloky 8-24 a ich uvoľnenie, potom alokovanie 450.

Tento test skúma, či po uvoľnení ostane celý blok do ktorého sa dá alokovať väčší blok.

Test prebehol úspešne.

TEST 3:

Pamäť 2000, náhodné bloky 300-1000 a ich uvoľnenie, potom alokovanie 1500.

Rovnako ako test 2, v tomto prípade oveľa väčšie hodnoty.

Funkčné aj s väčšími hodnotami.

TEST 4:

Pamäť 2000, náhodné bloky 8-500 a ich uvoľnenie.

Skúma, či dokáže pracovať s menšími a zároveň menšími hodnotami.

Taktiež nakonci mi ostal po úplnom uvoľnení celý voľný blok.

Martin Hric
ID:111696

Záver:

Časová a pamäťová zložitosť:

memory_init – časovo O(n), keďže celý blok nastavujem na 0.

memory_alloc – časovo O(n), je to implicitný list. Pamäťovo lepšie, keďže mám iba hlavičku a pätičku int.

memory_check – časovo O(1), tam je len jeden if.

memory_free – časovo O(1), to len prebehne pár výpočtov, hoci mohol som ušetriť čas keby som nenuloval.