システムソフトウェア　課題1

20B30100　伊藤悠馬

2022年11月2日

第1章　sys_freemem実装レポート

1.1　システムコール sys_freemem の作成方針，使用したアルゴリズム等

メモリ管理を行なっている kcalloc.c でカーネルのフリーメモリリストのリスト要素をカウントし、定数として宣言されている 1 ページあたりのバイト数をかけることをもって求める方針とした。

1.2　どのファイルを何のためにどう変更したか

1. kernel/kalloc.c

```
0                uint64
1                freemem(void){
2                uint64 free_bite = 0;// 空きメモリ (bite)
3                struct run *run;// 現在参照している空きメモリリスト
4
5                acquire(&kmem.lock);//  ロック
6                run = kmem.freelist;
7                while(run){
8                free_bite += PGSIZE;
9                run = run->next;
10               }
11               release(&kmem.lock);// ロック解除
12
13               return free_bite;
14               }
```

freemem() の定義本体。kmem をロックすることで他のプロセスがメモリの確保、解放を行なっている場合も正しく動作をすることができる。

メモリのページ数の計測は kmem.freelist の要素数をカウントすることで行い、それにページ一つのバイト数をかけることでメモリの空き容量 (bite) を求めている。

2. kernel/syscall.c

```
0                [SYS_mkdir] sys_mkdir,
1                [SYS_close] sys_close,
```

```
2                   [SYS_getppid] sys_getppid,
3                   [SYS_freemem] freemem,
```

システムコール番号と関数を結びつける配列に freemem を追加。

3. kernel/syscall.h

```
0                   #define SYS_freemem 23
```

システムコール番号を割り当てる。

4. user/freememtest.c

```
0                   #include "kernel/types.h"
1                   #include "user/user.h"
2                   int main() {
3                   printf("%l\n", freemem());
4                   sbrk(1);
5                   printf("%l\n", freemem());
6                   sbrk(4095);
7                   printf("%l\n", freemem());
8                   sbrk(1);
9                   printf("%l\n", freemem());
10                  sbrk(-4096);
11                  printf("%l\n", freemem());
12                  sbrk(-1);
13                  printf("%l\n", freemem());
14                  exit(0);
15                  }
```

freemem() をテストするための freemem() を使ったプログラム。

5. user/user.h

```
0                   int freemem(void);
```

user.h に宣言を記載。

6. user/usys.pl

```
0                   entry("freemem");
```

アセンブリ言語での各関数のラベル定義部分に freememm を追加

7. Makefile

```
0                   UPROGS=\
1                        ...
2                        $U/_freememtest\
```

makefile に freemem 用のテストを追加

1.3 テストプログラムの実行結果と，なぜそのような結果になったかの説明

1. 実行結果

```
0                   $ freememtest
1                        133382144
2                        133378048
3                        133378048
4                        133373952
5                        133378048
6                        133382144
```

2. 結果の考察今回のテストプログラムではsbrk() を使っているがこれは空きメモリから引数で指定したバイト数だけ確保するというものである。始めに空きメモリ容量を表示したのち、sbrk(1); で 1 バイトを確保する。しかしこの時ページ単位でメモリを確保する性質上、1 ページ 4096 バイトが一括で確保される。そのため 2 回目に空きメモリ容量を表示するタイミングでは始めの空きメモリ容量から 4096 バイト少ない値となっている。

   次にsbrk(4095); を実行した際には 4095 バイトが確保されるが、sbrk(1); で確保したページで賄えるため新たにメモリが確保されることはない。そのため 3 回目の空きメモリ容量表示では前回と同じになっている。

   次にsbrk(1); とすると空いているページがないため新しく 1 ページを確保することで、4 回目の空きメモリ容量表示では 4096 バイト減った値となっている。

   ここでsbrk(-4096); として 1 ページ分解放することで、空き容量が 1 ページ分増え、5 回目の空きメモリ容量表示では前回より 4096 バイト増えた値となっている。

   そしてsbrk(-1); とすることで上記で確保したメモリの最後の 1 バイトが解放され、最後の空きメモリ表示では最初と同じバイト数となっている。

1.4 使用したPC環境

1. CPU：1.4 GHz クアッドコア Intel Core i5
2. メモリ：8 GB 2133 MHz LPDDR3
3. ホスト OS：macOS Monterey バージョン 12.6
4. 実行環境：授業で使用している Docker

1.5　参考にした書籍やサイト (URL)

1. sbrk(2) manual page：https://nxmnpg.lemoda.net/2/sbrk
2. xv6: シンプルで Unix 風な教育用オペレーティングシステム:https://www.sugawara-lab.jp/fig/xv6-riscv-jpn.pdf

1.6　本課題によって得られた知見

LaTeX の環境構築をすることができ、tex の自動コンパイルを行うことができた。
カーネル用のメモリの使い方、ページについて理解が深められた。