CS342 Operating Systems, Spring 2012
Project 3: Memory Management

Assigned: 12.04.2012
Due date: 26.04.2012

Leap in and try things. If you succeed, you can have enormous influence. If you fail, you have still learned something, and your next attempt is sure to be better for it.. --Brian Kernighan

In this project, you will write a memory allocation module (a C source file) that can be used by a program to allocate memory dynamically. The module will have a functionality like the malloc library, but the amount of space allocated will be in kilobytes (KB). Additionally, the module will use the Buddy System algorithm to manage memory (a logical memory area). Read the Buddy System algorithm from the text book (it is in the virtual memory chapter). You can also find information in Internet.

The module you will write will be compiled together with a program that will use it. Hence it will be part of the address space of the program. This is a user level module (library), not a kernel module. It will be just a C file without a main() function and will be first compiled into an object file (the Makefile to do this is given to you below). The source file implementing your module will be called mem.c. The prototypes of the functions implemented there will be included in a header file called mem.h which will be included in an application program (for example app.c) that would like to use the module to allocate memory in multiples of KB.

The module will implement the following functions.

```
int  meminit(int totalmemsize); // totalmemsize is in KB
void *memalloc(int size);
void  memfree (void *ptr);
```

The meminit function is already implemented in the mem.c file below. You will implement memalloc and memfree functions. The meminit function is using the sbrk function (please read the related man page) to create a usable logical memory area in the address space of the program (namely it extends the data section of the program; you will manage this extended section). The global variable startptr points to the start of this memory area that will be managed by your code using the Buddy System algorithm. Note that you can not use the malloc function anymore since it is also trying to manage this area. Never use malloc in mem.c or app.c.

The maximum size of the are to be managed can be 1024 KB. The minimum size can be 16 KB. That means the min value of totalmemsize is 16 and max value is 1024. It has to be a power 2.

The skelaton of the mem.c file (in which you will do your implementation of the module) is shown below. You will modify and enhance this implementation.

mem.c

```
// This is the implementation of the memory allocation module

#include <stdio.h>
#include <unistd.h>
```

```
#include <stdlib.h>

#include "mm.h"

void *startptr = NULL; // points to the start of memory to manage


// extend the current data segment by totalmemsize
// so that we have memory to manage
// we will manage this memory
// - returns a pointer to the beginning of the memory to manage
// - parameter is in KB
int meminit(int totalmemsize)
{
      void * p;

      // read the man page of sbrk
      p = sbrk ( (intptr_t) totalmemsize * 1024);

      if (p == (void *) -1)
            return -1; // fail
      else {
            startptr = p;
            return 0; // success
      }
}


// returns NULL if allocation is not successful, otherwise
// returns a pointer to the allocated space
void *memalloc(int size)
{
      // use buddy algorithm to manage memory
}

// ptr points to the space allocated earlier and to be
// freed now
void  memfree (void *ptr)
{


}
```

The header file mem.h is shown below. You should not change this interface.

mem.h

```
// this is the interface of the memory allocation module (library)

int  meminit(int totalmemsize);

void *memalloc(int size);

void  memfree (void *ptr);
```

An application program that is using the module is shown below. You can write another program or modify this one to test your module.

app.c

```
// This is an application using the memory allocation module

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "mm.h"

int main()
{
      void *p1, *p2, *p3;
      int ret;

      ret = meminit (64); // 64 KB memory area to manage
      if (ret == -1) {
            printf ("could not create a memory area to manage");
            exit (1);
      }


      //...
      //...
      p1 = memalloc (10); // allocate 10 KB space
      if (p1 == NULL) {
            printf ("no more memory");
            exit (1);

      }
      // you can write something to the space pointed by p1

      p2 = memalloc (20); // allocate 20 KB space
      if (p1 == NULL) {
            printf ("no more memory");
            exit (1);
      }

      p3 = memalloc (5); // allocate 5 KB space
      if (p3 == NULL) {
            printf ("no more memory");
            exit (1);
      }


      //....
```

```
      memfree(p2);
      memfree(p1);
      memfree(p3);
}
```

A Makefile that can be used to compile the module and an application, and link them together is shown below.

Makefile

```
all: app  mem.o app.o

mem.o:       mem.c
           gcc -g -Wall -c  mem.c

app.o:       app.c
           gcc -g -Wall -c app.c

app: app.o mem.o
           gcc -o app mem.o app.o

clean:
      rm -fr *.o *~ app
```

You will submit your program online.