/\*

\* 9.14.c

\*/

#include <stdio.h>

#include <assert.h>

#include "vm/csapp.h"

void test(char\* filename, char\* content) {

int fd;

char buf[20];

fd = Open(filename, O\_RDONLY, 0);

Read(fd, buf, strlen(content));

assert( !strncmp(buf, content, strlen(content)) );

}

int touch(char\* filename, char\* content) {

int fd;

umask(DEF\_UMASK);

fd = Open(filename, O\_WRONLY|O\_CREAT|O\_TRUNC, DEF\_MODE);

Write(fd, content, strlen(content));

Close(fd);

}

int main(int argc, char\* argv[]) {

touch("hello.txt", "Hello, world!");

test("hello.txt", "Hello, world!");

struct stat stat;

int fd;

char\* bufp;

size\_t size;

fd = Open("hello.txt", O\_RDWR, 0);

fstat(fd, &stat);

size = stat.st\_size;

bufp = Mmap(NULL, size, PROT\_WRITE, MAP\_SHARED, fd, 0);

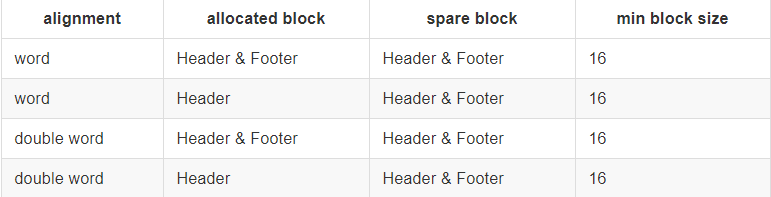
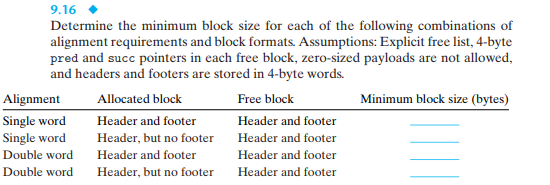
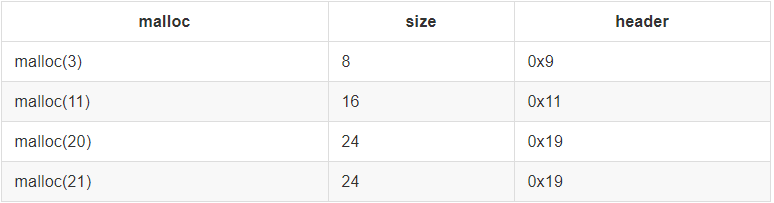
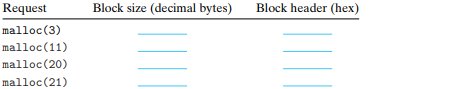
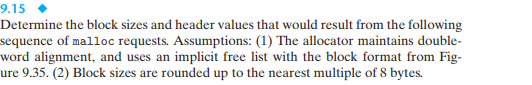
\*bufp = 'J';

Munmap(bufp, size);

test("hello.txt", "Jello, world!");

return 0;

}



--- mm.c 2017-11-09 02:57:43.679935907 +0000

+++ mm.9.17.c 2017-11-09 02:57:43.679935907 +0000

@@ -41,6 +41,7 @@

/\* Global variables \*/

static char \*heap\_listp = 0; /\* Pointer to first block \*/

+static char \*rover; /\* Next fit rover \*/

/\* Function prototypes for internal helper routines \*/

static void \*extend\_heap(size\_t words);

@@ -69,6 +70,7 @@

heap\_listp += (2\*WSIZE); //line:vm:mm:endinit

/\* $end mminit \*/

+ rover = heap\_listp;

/\* $begin mminit \*/

/\* Extend the empty heap with a free block of CHUNKSIZE bytes \*/

@@ -177,6 +179,10 @@

bp = PREV\_BLKP(bp);

}

/\* $end mmfree \*/

+ /\* Make sure the rover isn't pointing into the free block \*/

+ /\* that we just coalesced \*/

+ if ((rover > (char \*)bp) && (rover < NEXT\_BLKP(bp)))

+ rover = bp;

/\* $begin mmfree \*/

return bp;

}

@@ -290,16 +296,20 @@

{

/\* $end mmfirstfit \*/

- /\* $begin mmfirstfit \*/

- /\* First-fit search \*/

- void \*bp;

-

- for (bp = heap\_listp; GET\_SIZE(HDRP(bp)) > 0; bp = NEXT\_BLKP(bp)) {

- if (!GET\_ALLOC(HDRP(bp)) && (asize <= GET\_SIZE(HDRP(bp)))) {

- return bp;

- }

- }

- return NULL; /\* No fit \*/

+ /\* Next fit search \*/

+ char \*oldrover = rover;

+

+ /\* Search from the rover to the end of list \*/

+ for ( ; GET\_SIZE(HDRP(rover)) > 0; rover = NEXT\_BLKP(rover))

+ if (!GET\_ALLOC(HDRP(rover)) && (asize <= GET\_SIZE(HDRP(rover))))

+ return rover;

+

+ /\* search from start of list to old rover \*/

+ for (rover = heap\_listp; rover < oldrover; rover = NEXT\_BLKP(rover))

+ if (!GET\_ALLOC(HDRP(rover)) && (asize <= GET\_SIZE(HDRP(rover))))

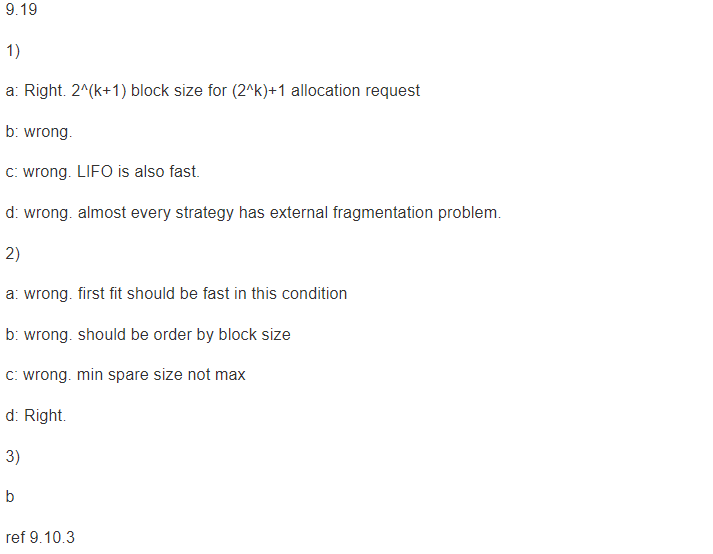
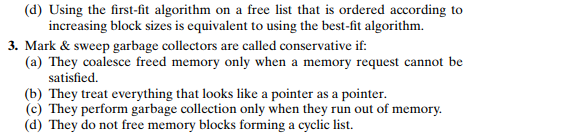
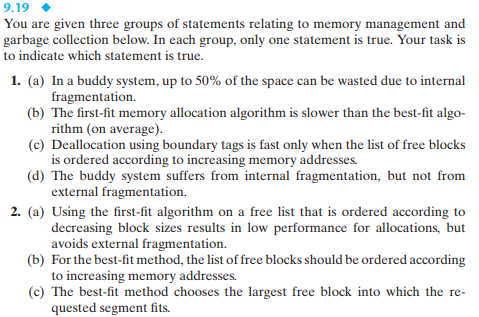
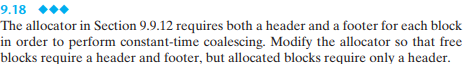
+ return rover;

+

+ return NULL; /\* no fit found \*/

}

/\* $end mmfirstfit \*/



using malloc lib code from csapp site

* implicit idle list
* allocated block with header & footer
* idle block with header & footer
* no GC
* first fit strategy

another modification to malloc lib file

mm.h

--- ../vm/mm.h 2017-11-09 02:57:43.679935907 +0000

+++ mm.h 2017-11-09 02:57:43.679935907 +0000

@@ -5,6 +5,9 @@

extern void mm\_free (void \*ptr);

/\* $end mallocinterface \*/

+#define malloc(size) mm\_malloc(size)

+#define free(ptr) mm\_free(ptr)

+

extern void \*mm\_realloc(void \*ptr, size\_t size);

extern void \*mm\_calloc (size\_t nmemb, size\_t size);

extern void mm\_checkheap(int verbose);

memlib.c

--- ../vm/memlib.c 2017-11-09 02:57:43.679935907 +0000

+++ memlib.c 2017-11-09 02:57:43.675935936 +0000

@@ -15,23 +15,18 @@

#include "csapp.h"

#include "memlib.h"

-#define MAX\_HEAP (20\*(1<<20)) /\* 20 MB \*/

-

-

/\* $begin memlib \*/

/\* Private global variables \*/

static char \*mem\_heap; /\* Points to first byte of heap \*/

static char \*mem\_brk; /\* Points to last byte of heap plus 1 \*/

-static char \*mem\_max\_addr; /\* Max legal heap addr plus 1\*/

/\*

\* mem\_init - Initialize the memory system model

\*/

void mem\_init(void)

{

- mem\_heap = (char \*)Malloc(MAX\_HEAP);

+ mem\_heap = (char \*)sbrk(0);

mem\_brk = (char \*)mem\_heap;

- mem\_max\_addr = (char \*)(mem\_heap + MAX\_HEAP);

}

/\*

@@ -43,7 +38,7 @@

{

char \*old\_brk = mem\_brk;

- if ( (incr < 0) || ((mem\_brk + incr) > mem\_max\_addr)) {

+ if ( (incr < 0) || ((mem\_brk = sbrk(incr)) == (void \*)-1)) {

errno = ENOMEM;

fprintf(stderr, "ERROR: mem\_sbrk failed. Ran out of memory...\n");

return (void \*)-1;

main.c file measure malloc performance

/\*

\* main.c

\*/

#include <stdio.h>

#include "csapp.h"

#ifdef CUS\_MALLOC

#include "mm.h"

#include "memlib.h"

#else

#include <stdlib.h>

#endif

#define LOOP 10000

int main(int argc, char\* argv[]) {

void\* m\_start = sbrk(0);

size\_t malloc\_size = 0;

int i;

for (i = 0; i < LOOP; i+=2) {

void\* ptr\_f = malloc(i);

void\* ptr = malloc(i+1);

free(ptr\_f);

malloc\_size += i+1;

}

void\* m\_end = sbrk(0);

size\_t heap\_size = (size\_t)(m\_end - m\_start);

printf("malloc size: %ld, heap\_size: %ld\n", malloc\_size, heap\_size);

return 0;

}

run make to generate both origin main executable file and custom version(using -DCUS\_MALLOC)

CC = gcc

CFLAGS = -m64 -pthread -DCUS\_MALLOC

SRCS = mm.c memlib.c csapp.c

all: origin custom diff

measure:

time ./origin.main

time ./custom.main

origin:

$(CC) -m64 main.c -o origin.main

custom:

$(CC) $(CFLAGS) $(SRCS) main.c -o custom.main

diff:

(diff -u ../vm/mm.h mm.h > mm.h.diff; cd .)

(diff -u ../vm/memlib.c memlib.c > memlib.c.diff; cd .)

test:

.PHONY: clean

clean:

find . -type f -executable -print0 | xargs -0 rm -f --

measurement

(cd chapter9/code/malloc; make measure)

time ./origin.main

malloc size: 25000000, heap\_size: 28311552

0.00user 0.01system 0:00.01elapsed 100%CPU (0avgtext+0avgdata 19256maxresident)k

0inputs+0outputs (0major+4547minor)pagefaults 0swaps

time ./custom.main

malloc size: 25000000, heap\_size: 31327104

0.58user 0.00system 0:00.58elapsed 99%CPU (0avgtext+0avgdata 30592maxresident)k

0inputs+0outputs (0major+7339minor)pagefaults 0swaps