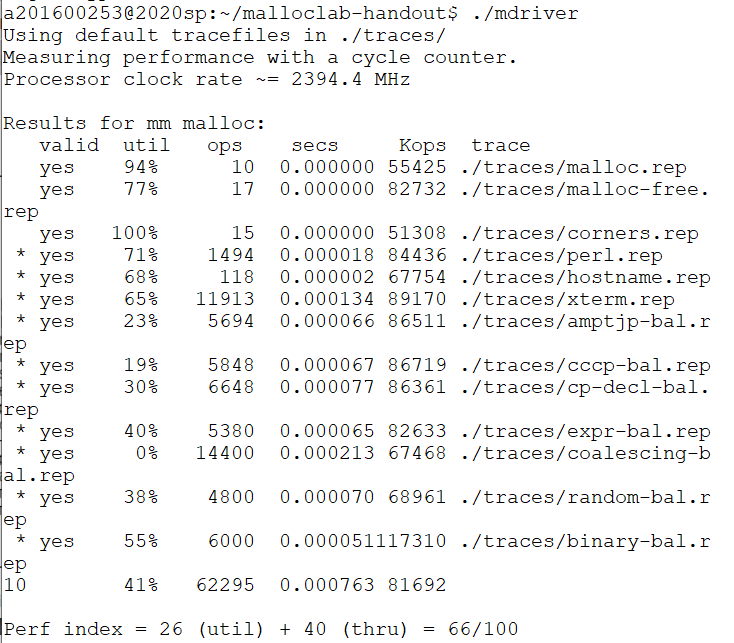
**2020 시스템 프로그래밍**

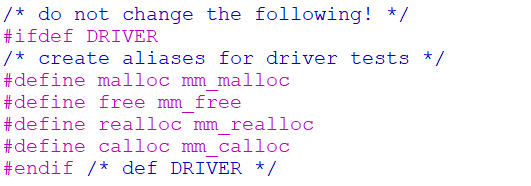
**- Lab 04 -**

|  |  |
| --- | --- |
| **제출일자** | 2020.12.8 |
| **분 반** | 00 |
| **이 름** | 신세정 |
| **학 번** | 201600253 |

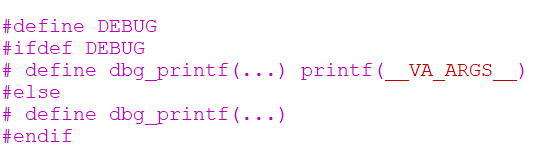
|  |
| --- |
| **Naïve** |



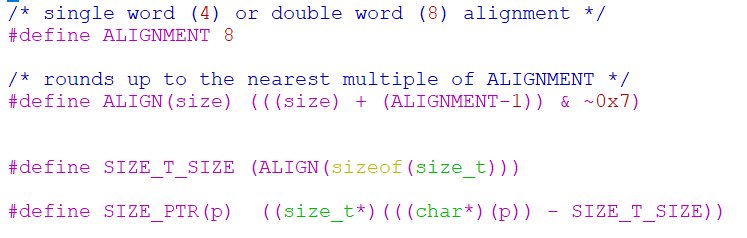
|  |
| --- |
| 구현 방법 |



DRIVER이 정의 되어있으면 위의 코드 실행한다.



DEBUG가 정의 되어있으면 위 dbg\_printf() printf(\_\_VA\_ARGS\_\_)문을 실행하고 아니면 dbg\_printf()을 실행한다



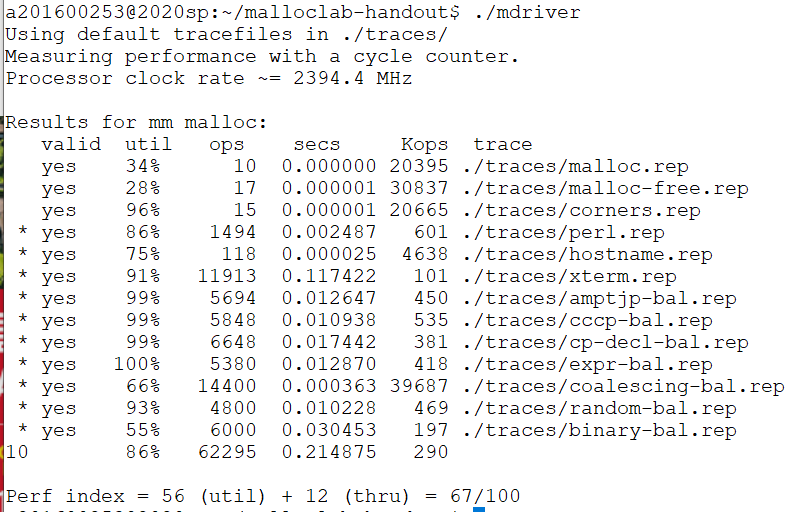
ALINEMENT 상수를 정의하고있는데 더블워드크기를 정의하고 있다.

ALING(size) 는 주어진 크기의 수에 가까운 ALIGNMENT배수 값으로 반올림한다

SIZE\_T\_SIZE는 주어진size\_t만큼 정렬하고 이 값을 SIZE\_T\_SIZE로 정의한다.

SIZE\_PRT는 주소p가 가리키는 블록에서 크기 정보가 저장되어 있는 부분의 주소를 반환한다.

|  |
| --- |
| **implicit** |



|  |
| --- |
| 구현 방법 |

#define WSIZE 4 : 상수를 정의하고있는데 워드크기를 정의하고 있다.

#define DSIZE 8 : 상수를 정의하고있는데 더블워드크기를 정의하고 있다.

#define CHUNKSIZE (1 << 12) : 초기 heap size 설정한다.

#define OVERHEAD 8 : header+footer사이즈, 실제 데이터 저장되는 공간이 아니라 overhead가 저장된다.

#define MAX(x, y)((x) > (y) ? (x) : (y)) : x와 y중 큰 값 구한다.

#define PACK(size, alloc) ((size) | (alloc)) : size와 alloc(z)의 값을 한 word로 묶는다

#define GET(p)(\*(unsigned int \*)(p)) : 포인터p가 가리키는 곳의 word값을 읽어온다.

#define PUT(p, val)(\*(unsigned int \*)(p) = (val)) : 포인터p가 가리키는 곳의 word값에 val을 기록한다.

#define GET\_SIZE(p) (GET(p) & (~0x7)) : 포인터p가 가리키는 곳에서 word를 읽어 하위 3bit를 버린다.

#define GET\_ALLOC(p) (GET(p) & 0x1) : 포인터p가 가리키는 곳에서 word를 읽어 최하위 1bit를 읽는다.

#define HDRP(bp)((char \*)(bp) - WSIZE) : 주어진 포인터bp의 header주소를 계산한다.

#define FTRP(bp) ((char\*)(bp) + GET\_SIZE(HDRP(bp)) - DSIZE) : 주어진 포인터 bp의 footer 주소를 계산한다.

#define NEXT\_BLKP(bp) ((char\*)(bp) +GET\_SIZE((char \* )(bp) -WSIZE)) : 주어진 포인터bp를 이용해 다음 블록 주소를 얻어온다.

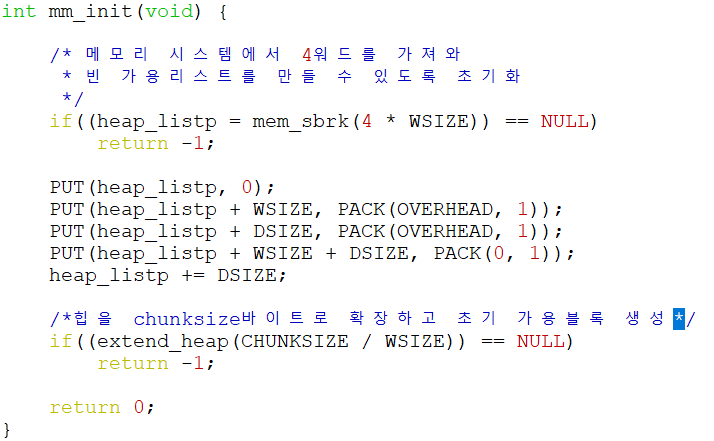
#define PREV\_BLKP(bp) ((char\*)(bp)- GET\_SIZE((char \* )(bp) -DSIZE)) : 주어진 포인터bp를 이용해 이전 블록 주소를 얻어온다.

1. int mm\_init(void)

할당기가 초기화를 완료하고 어플리케이션으로부터 할당과 반환 요청을 받을 준비를 완료한다. sbrk함수를 이용해 4워드를 할당해 빈 가용리스트를 만들 수 있도록 초기화 하고, 첫번쨰 워드에는 더블 워드 경계로 정렬된 미사용 패딩워드를 넣어준다.

그 다음 프롤로그 블록이 오는데 이는 header와 footer로만 구성된 8바이트 할당 블록이다. 프롤로그 블록은 초기화 과정에서 생성되며 절대 반환되지 않는다.

힙은 에필로그 블록으로 끝나고 이는 헤더로만 구성된 크기가 0으로 할당될 블록이다.

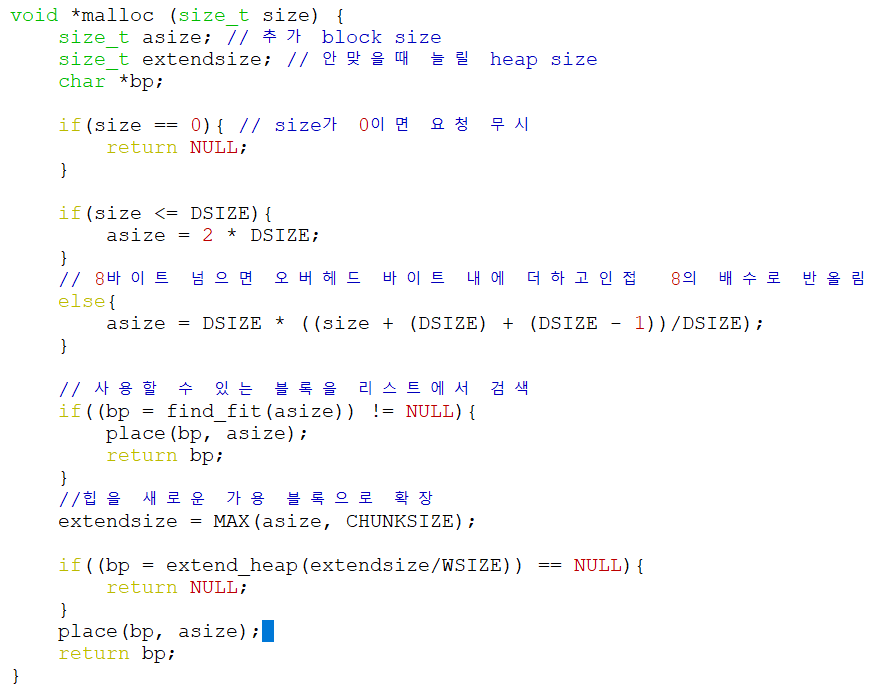


1. void \*malloc(size\_t size)

어플리케이션은 malloc함수를 호출해 size바이트 메모리 블록을 요청한다.

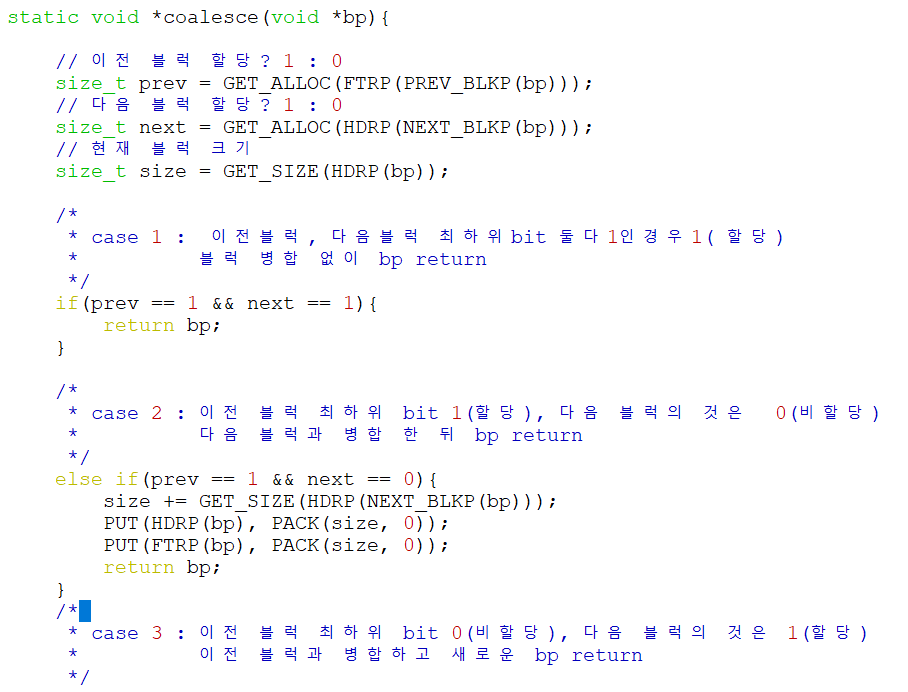
이러한 요청들을 확인한 뒤 할당기는 요구된 블록 크기를 조절해 header와 footer를 위한 공간을 확보하고, double word 조건을 만족한다. 8바이트는 정렬 조건을 만족시키기 위해 추가적인 8바이트는 header + footer를 위해 최소 16바이트 크기 블록을 구성할 수 있도록 한다.

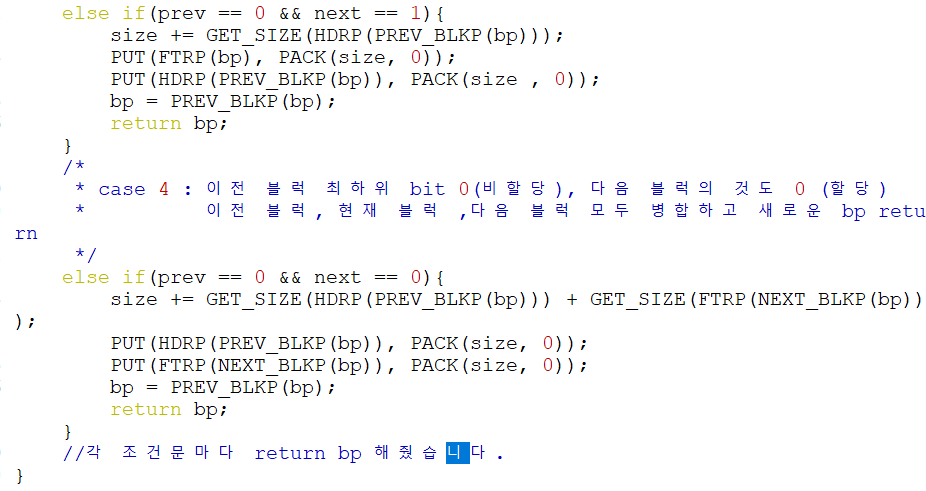
할당기가 요구한 크기를 조정한 뒤에 find\_fit을 이용해 적절한 가용 블록을 가용리스트에서 검색한다. 만약 맞는 블록을 찾으면 할당기는 place함수를 이용해 요청한 블록을 배치한다. 초과된 부분은 분할하고 새롭게 할당된 블록을 반환한다. 만약 이 과정에서 메모리가 부족하면 extend\_heap을 이용해 메모리 공간을 더 할당하고, 앞의 과정을 반복한다. 이후 새롭게 할당한 블록 포인터를 리턴한다.



1. static void \*coalesce(void \*bp)

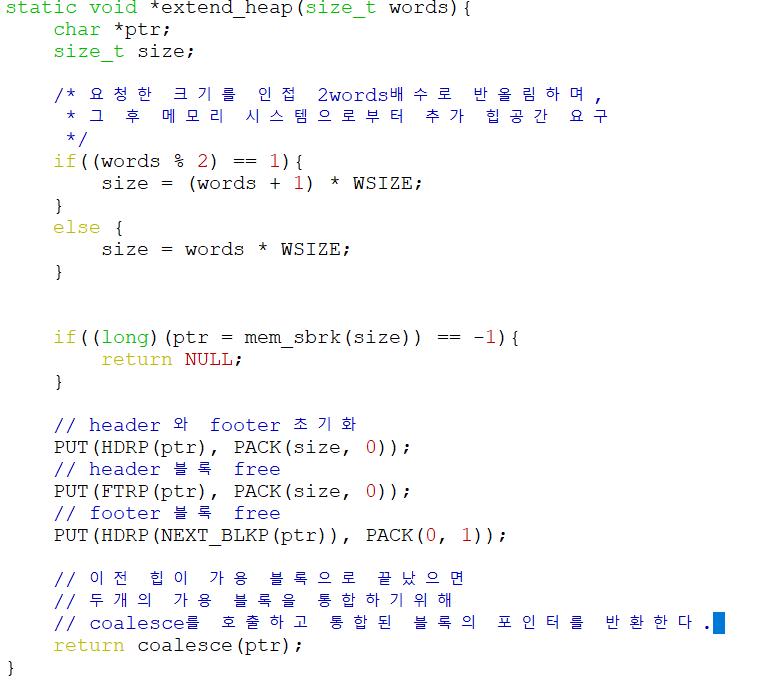
인접 가용 블록을 합치는 함수이다. 할당기가 현재 블록을 반환할 때 가능한경우는 4가지 이다. (주석으로 처리)



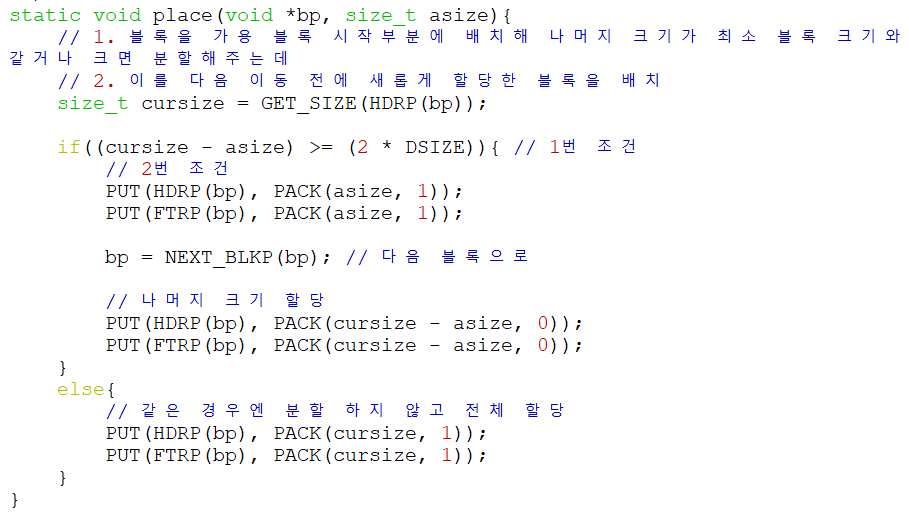


1. static void \*extend\_heap(size\_t words)

이 함수는 새 가용 블록으로 힙을 확장하는 함수이다.

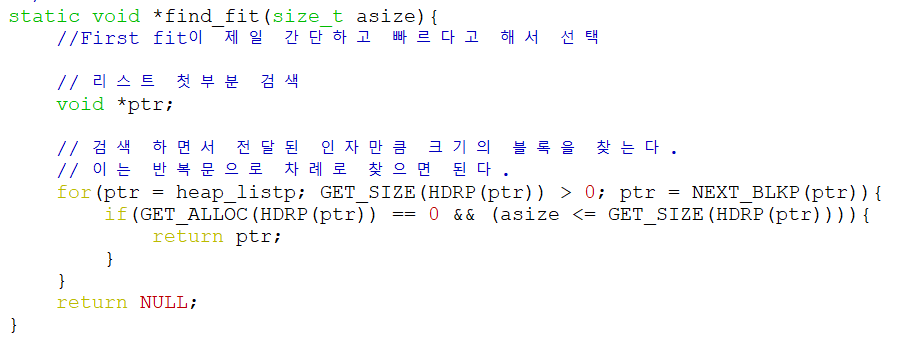


1. static void place(void \*bp, size\_t asize)



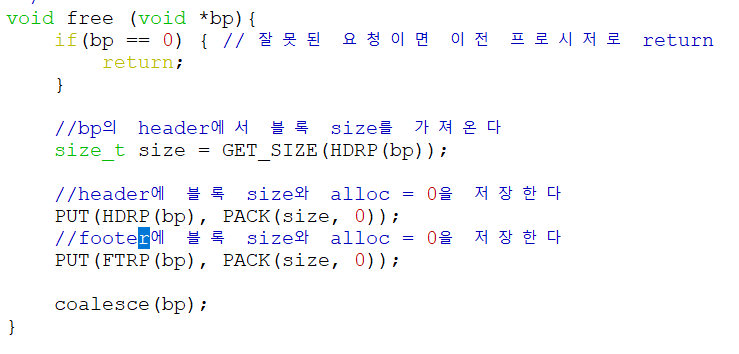
1. Static void \*find\_fit(size\_t asize)

First fit이 가장 빠르다고 하기에 이를 이용했다.



1. void free(void \*bp)

이전에 할당한 블록을 free함수를 통해 반환한다.



|  |
| --- |
| **explicit** |

해결하지 못했습니다.

|  |
| --- |
| 구현 방법 |

#define WSIZE 4 : 한 워드 크기

#define DSIZE 8 : 더블 워드 크기

#define CHUNKSIZE (1<<12) : 초기 힙 크기

#define MINIMUM 24 // 최소 블록 크기

#define OVERHEAD 8 : header + footer크기

#define MAX(x, y) ((x) > (y)? (x) : (y)) : max 크기

#define MIN(x, y) ((x) < (y)? (x) : (y)) : min 크기

#define PACK(size, alloc) ((unsigned)((size) | (alloc))) : 크기와 alloc비트를 합친다

#define GET(p) (\*(unsigned \*)(p)) : 주소 p가 가리키는 곳에 있는 값을 읽어온다.

#define PUT(p, val) (\*(unsigned \*)(p) = (unsigned)(val)) : 주소p에 값 val을 저장한다.

#define GET8(p) (\*(unsigned long \*)(p)) : 주소p가 가리키는 곳에 있는 값 8바이트를 읽어온다.

#define PUT8(p, val)(\*(unsigned long \*)(p) = (unsigned long)(val)) : 주소 p에 있는 값 val을 8바이트 저장한다.

#define GET\_SIZE(p) (GET(p) & (~0x7)) : 주소 p에 있는 크기를 읽어온다.

#define GET\_ALLOC(p) (GET(p) & 0x1) : 주소 p에 있는 allocated bit를 읽어온다.

#define HDRP(bp) ((void \*)(bp) - WSIZE) : 포인터 bp의 header를 계산한다.

#define FTRP(bp) ((void \*)(bp) + GET\_SIZE(HDRP(bp)) - DSIZE) : 포인터 bp의 footer를 계산한다.

#define NEXT\_BLKP(bp)((void \*)(bp) + GET\_SIZE(HDRP(bp))) : 블록 포인터 bp의 다음 블록을 계산한다.

#define PREV\_BLKP(bp)((void \*)(bp) - GET\_SIZE(HDRP(bp) - DSIZE)) : 블록 포인터 bp의 이전 블록을 계산한다.

#define NEXT\_FREEP(bp) (\*(void \*\*)(bp)) : 블록 포인터 bp의 다음 가용 블록을 계산한다.

#define PREV\_FREEP(bp) (\*(void \*\*)(bp) + DSIZE) : 블록 포인터 bp의 이전 가용 블록을 계산한다.

블록 포인터 bp를 free하고 다음, 이전 가용 블록을 계산한다.

#define NEXT\_FREE\_BLKP(bp) ((char \*)GET8(char \*)(bp))

#define PREV\_FREE\_BLKP(bp) ((char \*)GET8(char \*)(bp) + DSIZE)

#define NEXT\_BLKP(bp) ((char \*)(bp) + GET\_SIZE((char \*)(bp) - WSIZE))

#define PREV\_BLKP(bp) ((char \*)(bp) - GET\_SIZE((char \*)(bp) - DSIZE))