Bu atama malloc() fonksiyonu yazmak içindir. Her zaman olduğu gibi birkaç adım devam edeceğiz.

Bu Labı sunarken sadece son versiyonuna ihtiyacınız var. **(kısım 4 veya 5'ten** itibaren)

Dosyaları malloc.c ve burada(link) sağlanan Malloc.h sunacaktır (bu değişiklik yapmanıza gerek yoktur!) Bir test kaynak dosya test_malloc.c,(1)bir nesne dosyası malloc.o içine malloc.c ve (2)ismi test_malloc olan çalıştırabilir dosyanın içine test_malloc.c ve bir yapıdosyası derler

Emin olunuz ki rutin olarak yürütülen malloc() fonksiyonu , makefile tarafindan oluşturulan yalnızca yürütülebilir test_malloc olacak.

Bir başka şey ise kullanılan global değişkenler bu laboratuvar için gereklidir. Global değişkenleri dikkatle kullanın ve mümkün olduğunca küçük sayıları tutmaya çalışın!

KISIM 1: jmalloc1()

Herhangi bir şey yapmadan önce sana malloc(),ve sbrk() hakkında bilgi verecek bu linki (http://web.eecs.utk.edu/~plank/plank/classes/cs360/360/notes/Malloc1/lecture.html) incele.

Şimdi , jmalloc1() fonksiyonunu malloc() fonksiyonu gibi yazılabilir. Kullanıcı en az boyutta bir geri değer döndürür. Bu değer için tamponda boş bir bellek tahsis edilecek. Her hangi bir şey yapmak istemiyorsak jfree1() fonksiyonunu, yeni bir bellek alanını tahsis etmek için sbrk() fonksiyonu kullanın.

jmalloc1 () fonsiyonun , malloc () fonksiyonu gibi olması için eksra 8 bayt'a ihtiyaç vardır. bu 8 bayt ın dördü tamsayı olmalıdır. jmalloc1 () fonsiyonu için bellekte ter ayrılmıştır.

Eğer jmalloc1() fonksiyonunu kullanmak istemiyorsanız NULL döndürmelisiniz.

KISIM 2: Porting jmalloc1() to be malloc

jmalloc1() fonksiyonu çalıştırıldığında malloc.c ı aktarılmıştır. malloc(), free(), calloc() ve realloc() fonksiyonları bu dosyayi tanımlamalıdır .Bunu bir kere çalıştırdığınızda diğer programlarla bağlantılanabilir, ve normal standartlaşmış

unix kaynakları yerine malloc(), free(), calloc() ve realloc() fonksiyonlarını kullanir. malloc() fonksiyonu sadece jmalloc1() gibi olmalıdır. free() fonksiyonunun malloc() foksiyonunu kullanan programlar unix e göre daha fazla belleği kullanacak. calloc() ve realloc () fonksiyonları malloc() fonksiyonu ile uygulanabilir. başka bir deyiş ile calloc() a malloc() hatta bazen bzero() denebilir. bazen malloc() fonksiyonuna da bcopy() ve free() denebilir.

jmalloc1() fonksiyonunu malloc() fonksiyonu ile kullanmalısınız. Örneğin eğer printf() diye derlemeye çalışırsanız printf() fonksiyonuna malloc() denebileceğini unutmamalısınız.

KISIM 3: jmalloc2() and jfree2()

Hafızaya jmalloc2() yazmak istediğimizde aslında jfree2() diye çalışabilir. jmalloc2 () fonksiyonunu kullanacağınızda bir malloc uygulama dokumanına ihtiyaciniz var açıklama olarak bu linke (http://web.eecs.utk.edu/~plank/plank/classes/cs360/360/notes/Malloc2/lecture.html) başvurunuz. Eğer bu notları daha önce okumadıysanız jmalloc2() ve jfree2() uygulamaları bittiğinde bir liste kullanır. jmalloc() fonksiyonu kullanıcı için hafızanın bir kısmını alır. Eğer bu kısım önceden alındıysa kalan kısım için doküman incelenmelidir. jfree2 () listede hafızada kapladığı alan bellidir. Bu zor bir şeydir. onu programlarken dikkatli olmak gerekir.

KISIM 4: Porting jmalloc2() to be malloc

2. bölümünde olduğu gibi Free (), calloc () ve yeni işlevleri kullanmak için malloc.c içinde realloc () malloc () malloc.c ve uygulamak jmalloc2.c içeriğini kopyalayın. Daha önceki atamaları kodu ile çalıştığını gösterin. Yine çok dikkatli olmak gerekir bunu asla unutmayın.

KISIM 5: Optional

Sadece isterseniz bunu yaparsınız. Bu parça için kredi almayacaksınız. Bununla birlikte sana gerçek hack uygulaması verecek. Boş listede birleşen düğümler uygulanır. Diğer bir deyişle kullanıcı free() çağırdığında, boş listede belleğe yeni bırakılmış yığını diğer yığınlarla bitiştirerek bir öbek içinde bir araya getirir. Bu sınıf anlatıldığı gibi parçalanma engellemeye yardımcı olacaktır.