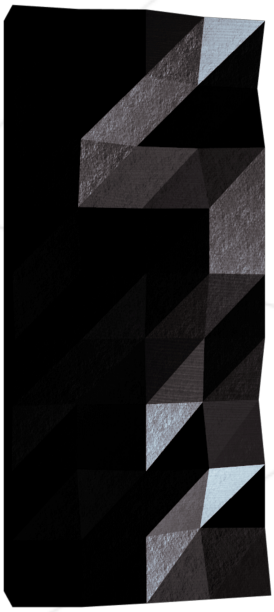

Libft

Sizin yazacağınız ilk kütüphane

Özet:
Bu projenin amacı bir `C` kütüphanesi yazmaktır.
Yazacağınız diğer programlarda işinize yarayacak yaygın kullanımlı fonksiyonları içerecektir.

Versiyon: 16.1

İçindekiler

Bölüm I

Giriş

`C` programlama, kullanışlı standart fonksiyonlara erişiminiz olmadığında çok sıkıcı olabilir. Bu proje, bu fonksiyonların çalışma mantığını anlamak, onları implement edebilmek ve nasıl kullanılması gerekteğini öğrenmekle ilgilidir. Kendi kütüphanenizi yaratacaksınız. Sonraki `C` projelerinde kullanabileceğiniz için, yarattığınız kütüphane faydalı olacaktır.

`Libft` kütüphanenizi yıl boyunca genişletmek için zaman ayırın. Ancak, yeni bir proje üzerinde çalışırken, kütüphanenize eklediğiniz fonksiyonlara proje yönergelerinde izin verildiğinden emin olmayı unutmayın.

Bölüm II

Genel Talimatlar

- Projeleriniz C programlama dilinde yazılmalıdır.
- Projeleriniz Norm'a uygun olarak yazılmalıdır. Bonus dosyalarınız/fonksiyonlarınız varsa, bunlar norm kontrolüne dahil edilir ve bu dosyalarda norm hatası varsa `0` alırsınız.
- Tanımlanmamış davranışlar dışında sizin fonksiyonlarınız beklenmedik bir şekilde sonlanmamalıdır (Segmentasyon hatası, bus hatası, double free hatası, vb.) . Eğer bunlar yaşanırsa s `0` alırsınız.
- Heap'de ayırmış olduğunuz hafıza adresleri gerekli olduğu durumlarda serbest bırakılmalıdır. Hiçbir istisna tolere edilmeyecektir.
- Eğer verilen görev `Makefile` dosyasının yüklenmesini istiyorsa, sizin kaynak dosyalarınızı `-Wall`, `-Wextra` , `-Werror`, flaglarini kullanarak derleyip çıktı dosyalarını üretecek olan Makefile dosyasını oluşturmanız gerekmektedir. Makefile dosyasını oluştururken cc kullanın ve Makefile dosyanız yeniden ilişkilendirme yapmamalıdır (relink).
- `Makefile` dosyanız en azından `$(NAME)`, `all`, `clean`, `fclean` ve `re` kurallarını içermelidir.
- Projenize bonusu dahil etmek için Makefile dosyanıza `bonus` kuralını dahil etmeniz gerekmektedir. Bonus kuralının dahil edilmesi bu projenin ana kısmında kullanılması yasak olan bazı header dosyaları, kütüphaneler ve fonksiyonların eklenmesini sağlayacaktır. Eğer projede farklı bir tanımlama yapılmamışsa, bonus projeleri `_bonus.{c/h}` dosyaları içerisinde olmalıdır. Ana proje ve bonus proje değerlendirmeleri ayrı ayrı gerçekleştirilmektedir.
- Eğer projeniz kendi yazmış olduğunuz `libft` kütüphanesini kullanmanıza izin veriyorsa, bu kütüphane ve ilişkili `Makefile` dosyasını proje dizinindeki `libft` klasörüne ilişkili `Makefile` dosyası ile kopyalamanız gerekmektedir. Projenizin `Makefile` dosyası öncelikle `libft` kütüphanesini kütüphanenin `Makefile` dosyasını kullanarak derlemeli ardından projeyi derlemelidir.
- Test programları sisteme yüklenmek zorunda değildir ve puanlandırılmayacaktır. Buna rağmen test programları yazmanızı şiddetle önermekteyiz. Test programları

sayesinde kendinizin ve arkadaşlarınız projelerinin çıktılarını kolaylıkla gözlemleyebilirsiniz. Bu test dosyalarından özellikle savunma sürecinde çok faydalanacaksınız. Savunma sürecinde kendi projeleriniz ve arkadaşlarınızın projeleri için test programlarını kullanmakta özgürsünüz.

- Çalışmalarınız atanmış olan git repolarına yüklemeniz gerekmektedir. Sadece git reposu içerisindeki çalışmalar notlandırılacaktır. Eğer Deepthought sizin çalışmanızı değerlendirmek için atanmışsa, bu değerlendirmeyi arkadaşlarınızın sizin projenizi değerlendirmesinden sonra gerçekleştirecektir. Eğer Deepthought değerlendirme sürecinde herhangi bir hata ile karşılaşılırsa değerlendirme durdurulacaktır.

Bölüm III

Zorunlu Bölüm

Program adı	`libft.a`
Teslim edilecek dosyalar	`Makefile, libft.h, ft_*.c`
Makefile	`NAME, all, clean, fclean, re`
Harici fonksiyonlar.	`Detaylar aşağıdadır`
Libft kullanılabilir mi?	`n/a`
Açıklama	`Kendi kütüphanenizi yazın: eğitiminiz boyunca faydalı bir araç olacak fonskiyonlar koleksiyonu.`

III.1 Teknik Kısıtlamalar

- Global değişken tanımlamak yasaktır.
- Kompleks bir fonksiyonu bölmek için yardımcı fonksiyonlara ihtiyacınız varsa, onları `static` olarak tanımlayın. Bu şekilde fonksiyonların scopeları kullanılan dosya ile sınırlandırılacaktır.
- Tüm dosyalarınızı repositorynizin root/kök dizinine yerleştirin.
- Kullanmadığınız dosyaları yüklemek yasaktır.
- Her `.c` dosyası, `-Wall -Wextra -Werror` compiler flagleri ile sorunsuz derlenebiliyor olmalıdır.
- Kütüphanenizi yaratmak için `ar` komutunu kullanmalısınız. `libtool` komutunun kullanılması yasaktır.
- `libft.a` dosyanız, repositorynizin rootunda/kökünde yaratılmalıdır.

III.2 Bölüm 1 - Libc Fonksiyonları

Başlangıç olarak, `libc`'den bir dizi fonksiyonu yeniden yazmalısınız. Yazdığınız fonksiyonlarınız orijinal versiyonları ile aynı prototiplere sahip olmalı ve aynı şekilde davranmalıdır. Orijinal fonksiyonun `man` sayfasındaki tanımına uyulması gerekmektedir. Yarattığınız fonksiyonun tek farkı ismi olmalıdır. Sizin fonksiyonlarınız '`ft_`' önekiyle başlayacaktır. Örneğin, `strlen` fonksiyonu, sizin `ft_strlen` fonksiyonunuz olacaktır.

Baştan yazmanız gereken bazı foksiyon prototipleri 'restrict' keywordünü kullanır. Bu keyword, c99 standardının bir parçasıdır. Bu nedenle, bu keywordü kendi fonksiyon prototiplerinize dahil etmeniz ve kodunuzu -std=c99 flagiyle derlemeniz yasaktır.

Aşağıdaki standart fonksiyonları kendi fonksiyonlarınız olarak baştan yazmanız gerekmektedir. Bu fonksiyonlar çalışmak için herhangi bir harici fonksiyona ihtiyaç duymamaktadır:

- `isalpha`
- `isdigit`
- `isalnum`
- `isascii`
- `isprint`
- `strlen`
- `memset`
- `bzero`
- `memcpy`
- `memmove`
- `strlcpy`
- `strlcat`
- `toupper`
- `tolower`
- `strchr`
- `strrchr`
- `strncmp`
- `memchr`
- `memcmp`
- `strnstr`
- `atoi`

Aşağıdaki iki fonksiyonu yazabilmek için `malloc()` kullanmanız gerekir:

- `calloc`
- `strdup`

III.3 Bölüm 2 - Ek Fonksiyonlar

İkinci bölümde, ya `libc`'de olmayan ya da farklı bir formda `libc`'nin bir parçası olan fonksiyonları geliştirmelisiniz.

Yukarıda kodladığınız fonksiyonlardan bazıları, 2. bölümünün fonsiyonlarını yazmak için yararlı olabilir.

Fonksiyon adı	`ft_substr`
Prototip	`char *ft_substr(char const *s, unsigned int start, size_t len);`
Teslim edilecek dosyalar	-
Parametreler	`s:` Substringin oluşturalacağı string. `start:` Substringin ana string içerisindeki başlangıç indeksi. `len:` Substringin maksimum uzunluğu.
Return değeri	Substring. Eğer allocation hatası varsa NULL döndürür.
Harici fonksiyonlar	`malloc`
Açıklama	Hafızada yer ayırır (malloc(3) ile) ve 's' stringinden bir substring return eder. Substring 'start' indeksinde başlar ve maksimum boyu 'len' dir.

Fonksiyon adı	`ft_strjoin`
Prototip	`char *ft_strjoin(char const *s1, char const *s2);`
Teslim edilecek dosyalar	-
Parametreler	`s1:` Ön string. `s2:` Son string.
Return değeri	Yeni oluşturulan string. Eğer allocation hatası varsa NULL döndürür.
Harici fonksiyonlar	`malloc`
Açıklama	Hafızada yer ayırır (malloc(3) ile), 's1' ve 's2' nin birleşimi olan yeni bir string return eder.

Fonksiyon adı	ft_strtrim
Prototip	char *ft_strtrim(char const *s1, char const *set);
Teslim edilecek dosyalar	-
Parametreler	s1: Kırpılacak string. set: Kırpılması istenen karakterler.
Return değeri	Kırpılmış string. Eğer allocation hatası varsa NULL döndürür.
Harici fonksiyonlar	malloc
Açıklama	Hafızada yer ayırır (malloc(3) ile), 's1' stringinin başından ve sonundan 'set' içerisindeki karakterleri çıkararak 's1' stringinin bir kopyasını döndürür.

Fonksiyon adı	ft_split
Prototip	char **ft_split(char const *s, char c);
Teslim edilecek dosyalar	-
Parametreler	s: Bölünecek string. c: Ayırıcı karakter.
Return değeri	Bölünme sonucu elde edilen string dizisi. Eğer allocation hatası varsa NULL döndürür.
Harici fonksiyonlar	malloc, free
Açıklama	Hafızada yer ayırır (malloc(3) ile), 's' stringini 'c' ayırıcı karakterine göre bölerek yeni bir string arrayi döndürür. String arrayi bir NULL pointer ile sonlanmalıdır.

Fonksiyon adı	ft_itoa
Prototip	char *ft_itoa(int n);
Teslim edilecek dosyalar	-
Parametreler	n: Dönüştürülecek olan integer değeri.
Return değeri	Verilen integer değerin string karşılığı. Eğer allocation hatası varsa NULL döndürür.
Harici fonksiyonlar	malloc
Açıklama	Hafızada yer ayırır(malloc(3) ile) ve integer olarak alınan değerinin string karşılığı döndürülür. Negatif sayılar da handle edilmelidir.

Fonksiyon adı	`ft_strmapi`
Prototip	`char *ft_strmapi(char const *s, char (*f)(unsigned int, char));`
Teslim edilecek dosyalar	-
Parametreler	`s: Iterate edilecek string.` `f: Her karaktere uygulanacak fonksiyon.`
Return değeri	`'f' fonksiyonun karakterlere uygulanması sonucu yaratılan string.` `Eğer allocation hatası varsa NULL döndürür.`
Harici fonksiyonlar	`malloc`
Açıklama	`Stringteki indeksini ilk argüman olarak ve karakterin kendisini ikinci argüman olarak göndererek, f fonksiyonunu s stringinin bütün karakterlerine uygular. f fonksiyonunun ardışık çalışmalarından yeni bir string yaratır (malloc(3) ile).`

Fonksiyon adı	`ft_striteri`
Prototip	`void ft_striteri(char *s, void (*f)(unsigned int, char*));`
Teslim edilecek dosyalar	-
Parametreler	`s: Iterate edilecek string.` `f: Her karatere uygulanacak fonksiyon.`
Return değeri	`Yok`
Harici fonksiyonlar	`Yok`
Açıklama	`Karakterin indeksini ilk argüman olarak göndererek, 'f' fonksiyonunu 's' stringinin bütün karakterlerine uygular. Her karakterin adresi, değiştirilebilme ihtimali doğrultusunda 'f' fonksiyonuna gönderilmelidir.`

Fonksiyon adı	`ft_putchar_fd`
Prototip	`void ft_putchar_fd(char c, int fd);`
Teslim edilecek dosyalar	-
Parametreler	`c: Çıktısı verilecek karakter.` `fd: Karakter çıktısının yazılacağı file descriptor.`
Return değeri	`Yok`
Harici fonksiyonlar	`write`
Açıklama	`File descriptora 'c' değişkeninin çıktısını verir.`

Fonksiyon adı	`ft_putstr_fd`
Prototip	`void ft_putstr_fd(char *s, int fd);`
Teslim edilecek dosyalar	-
Parametreler	`s: Çıktısı verilecek string.` `fd: String çıktısının yazılacağı file descriptor.`
Return değeri	`Yok`
Harici fonksiyonlar	`write`
Açıklama	`File descriptora 's' değişkeninin çıktısını verir.`

Fonksiyon adı	`ft_putendl_fd`
Prototip	`void ft_putendl_fd(char *s, int fd);`
Teslim edilecek dosyalar	-
Parametreler	`s: Çıktısı verilecek string.` `fd: String çıktısının ve yeni satırın yazılacağı file descriptor.`
Return değeri	`Yok`
Harici fonksiyonlar	`write`
Açıklama	`File descriptora 's' değişkeninin çıktısını verir, sonuna yeni satır ekler.`

Fonksiyon adı	`ft_putnbr_fd`
Prototip	`void ft_putnbr_fd(int n, int fd);`
Teslim edilecek dosyalar	-
Parametreler	`n: Çıktısı verilecek integer.` `fd: Integer çıktısının yazılacağı file descriptor.`
Return değeri	`Yok`
Harici fonksiyonlar	`write`
Açıklama	`File desciptora 'n' değişkeninin çıktısını verir.`

Bölüm IV

Bonus Bölüm

Zorunlu bölümü tamamladıysanız, daha ileri giderek bu ekstra bölümü de tamamlamaktan çekinmeyin. Bonus bölüm başarıyla geçilirse bonus puan kazandırır.

Hafıza ve stringleri manipüle etmeye yarayan fonksiyonlar çok kullanışlıdır. Ancak birazdan listeleri manipüle edebilen fonksiyonların daha da yararlı olduğunu keşfedeceksiniz.

Listenizin bir nodeunu temsil etmek için aşağıdaki yapıyı kullanmalısınız. Aşağıdaki declaration'ı `libft.h` dosyanıza ekleyin:

```
typedef struct      s_list
{
  void              *content;
  struct s_list     *next;
}                   t_list;
```

Aşağıda `t_list` struct'ının içindeki memberların tanımı yapılmaktadır:

- `content` : Node içerisindeki veri. `void *` herhangi türdeki veriyi tutmanızı sağlar.
- `next`: Bir sonraki nodeun adresini tutar. Eğer zaten son node ise `NULL` değerindedir.

Bonus fonksiyonlarını `libft.a` arşivine eklemek için Makefile dosyanıza `make bonus` kuralı ekleyin.

Bonus bölüm, yalnızca zorunlu bölüm KUSURSUZ ise değerlendirilecektir. Kusursuz, zorunlu bölümün tamamen yapıldığı ve sorunsuz çalıştığı anlamına gelir. TÜM zorunlu gereksinimleri tamamlamadıysanız, bonus bölüm hiçbir şekilde değerlendirilmeyecektir.

Listeleri kolaylıkla manipüle etmenizi sağlayacak aşağıdaki fonksiyonları implement edin.

Fonksiyon adı	`ft_lstnew`
Prototip	`t_list *ft_lstnew(void *content);`
Teslim edilecek dosyalar	-
Parametreler	`content: Yeni node'u oluşturacağınız içerik.`
Return değeri	`Yeni node`
Harici fonksiyonlar	`malloc`
Açıklama	`Hafızada yer ayırır(malloc(3) ile) ve yeni bir node return eder. Yeni nodeun 'content' member değişkeni fonksiyonun 'content' parametresi ile initialize edilir. 'Next' değişkeni ise NULL değeri ile initialize edilir.`

Fonksiyon adı	`ft_lstadd_front`
Prototip	`void ft_lstadd_front(t_list **lst, t_list *new);`
Teslim edilecek dosyalar	-
Parametreler	`lst: Listenin ilk node'unun pointerının adresi.` `new: Listeye eklenecek olan node'un adresi.`
Return değeri	`Yok`
Harici fonksiyonlar	`Yok`
Açıklama	`Listenin başına yeni bir 'new' node'u ekler.`

Fonksiyon adı	`ft_lstsize`
Prototip	`int ft_lstsize(t_list *lst);`
Teslim edilecek dosyalar	-
Parametreler	`lst: Listenin başlangıcı.`
Return değeri	`Listenin uzunluğunu döndürür.`
Harici fonksiyonlar	`Yok`
Açıklama	`Listedeki node sayısını sayar.`

Fonksiyon adı	`ft_lstlast`
Prototip	`t_list *ft_lstlast(t_list *lst);`
Teslim edilecek dosyalar	-
Parametreler	`lst: Listenin başlangıcı.`
Return değeri	`Listenin son node'u`
Harici fonksiyonlar	`Yok`
Açıklama	`Listenin son node'unu döndürür.`

Fonksiyon adı	`ft_lstadd_back`
Prototip	`void ft_lstadd_back(t_list **lst, t_list *new);`
Teslim edilecek dosyalar	-
Parametreler	`lst: Listenin ilk node'unun pointerının adresi.` `new: Listeye eklenecek olan node'un adresi.`
Return değeri	`Yok`
Harici fonksiyonlar	`Yok`
Açıklama	`Listenin sonuna yeni bir 'new' node'u ekler.`

Fonksiyon adı	`ft_lstdelone`
Prototip	`void ft_lstdelone(t_list *lst, void (*del)(void *));`
Teslim edilecek dosyalar	-
Parametreler	`lst: Free edilecek node.` `del: İçeriği silmek için kullanılacak fonksiyonun adresi.`
Return değeri	`Yok`
Harici fonksiyonlar	`free`
Açıklama	`Parametre olarak bir node alır ve parametre olarak verilen 'del' fonksiyonunu kullanarak node'un 'content' member değişkeninin kapladığı hafızayı ve node'un kendisini freeler. 'Next' member değişkeninin hafızası free edilmemelidir.`

Fonksiyon adı	`ft_lstclear`
Prototip	`void ft_lstclear(t_list **lst, void (*del)(void *));`
Teslim edilecek dosyalar	-
Parametreler	`lst: Listedeki herhangi bir node'un pointerının adresi.` `del: Node'un 'content' member değişkenini silmek için kullanılacak fonksiyonun adresi.`
Return değeri	`Yok`
Harici fonksiyonlar	`free`
Açıklama	`'del' ve free(3) kullanarak node'u ve ona bağlı olan bütün nodeları siler ve hafızadaki yerlerini temizler. Son olarak listenin pointerı NULL' a set edilmelidir.`

Fonksiyon adı	`ft_lstiter`
Prototip	`void ft_lstiter(t_list *lst, void (*f)(void *));`
Teslim edilecek dosyalar	-
Parametreler	`lst: Bir node'un adresi.` `f: Listenin içerisinde gezinip uygulanacak olan fonksyionun adresi.`
Return değeri	`Yok`
Harici fonksiyonlar	`Yok`
Açıklama	`Listeyi iterate eder ve 'f' fonksiyonunu listenin her node'unun 'content' member değişkenine uygular.`

Fonksiyon adı	ft_lstmap
Prototip	t_list *ft_lstmap(t_list *lst, void *(*f)(void *), void (*del)(void *));
Teslim edilecek dosyalar	-
Parametreler	lst: Bir node'un adresi. f: Listenin içerisinde gezinmek için kullanılacak olan fonksiyonun adresi. del: Gerekli olduğunda node'un 'content' ini temizlemeye yardımcı olan fonksiyonun adresi.
Return değeri	Yeni liste. Eğer allocation hatası varsa NULL döndürür.
Harici fonksiyonlar	malloc, free
Açıklama	'lst' listesini iterate eder ve 'f' fonksiyonunu listenin her node'unun 'content' member değişkenine uygular. Uygulama sonucunda oluşan yeni nodelardan yeni bir liste oluşturulur. Gerekli olduğu durumlarda 'del' fonksiyonu kullanılarak node'un 'content'i temizlenebilir.

Bölüm V

Proje Teslimi ve Akran Değerlendirmesi

Projenizi her zamanki gibi `Git` repositorynize gönderin. Savunma sırasında yalnızca reponuzdaki çalışmalar değerlendirilecektir. Dosyalarınızın adlarının doğru olduklarından emin olmak için adlarını iki kez kontrol etmekten çekinmeyin.

Tüm dosyalarınızı reponuzun root/kök dizinine yerleştirin.

```
Rnpu cebwrpg bs gur 97 Pbzzba Pber pbagnvaf na rapbqrq uvag.  Sbe
rnpu pvepyr, bayl bar cebwrpg cebivqrf gur pbeerpg uvag arrqrq sbe
gur arkg pvepyr.  Guvf punyyratr vf vaqvivqhny, gurer vf bayl n
cevmr sbe bar fghqrag jvaare cebivqvat nyy qrpbqrq zrffntrf.  Nal
nqinagntrq crbcyr pna cynl, yvxr pheerag be sbezre fgnss, ohg gur
cevmr jvyy erznva flzobyvp.  Gur uvag sbe guvf svefg cebwrpg vf:
Ynetr pbjf trarebfvgl pbzrf jvgu punegf naq sbhe oybaqr ungf gb qrsl
hccre tenivgl ureb
```