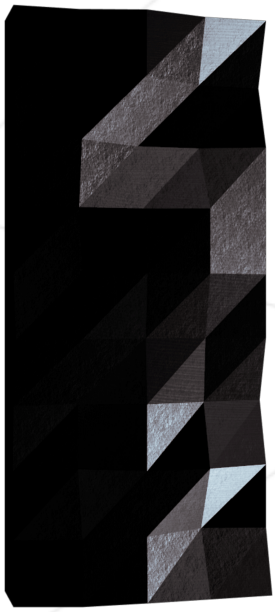

Libft

Kendi yazdığınız ilk kütüphane

Özet:
Bu projenin amacı bir `C` kütüphanesi yazmaktır.
Birçok genel amaçlı ve programlarınızın temel ablabiliceği fonksiyon içerecektir.

Versiyon: 15

İçindekiler

Bölüm I

Giriş

Son derece kullanışlı standart fonksiyonlara erişiminiz olmadığında `C` programlama çok sıkıcı olabilir. Bu proje, bu fonksiyonların çalışma mantığını anlamak, onları uygulamak ve nasıl kullanılması gerekteğini öğrenmekle ilgilidir.Kendi kütüphanenizi oluşturacaksınız. Bir sonraki `C` projelerinde kullanacağınız için, kütüphane faydalı olacaktır.

`Libft` yıl boyunca genişletmek için zaman ayırın. Ancak, yeni bir proje üzerinde çalışırken, kitaplığınızda kullanılan işlevlere 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 Kısım

Program adı	`libft.a`
Teslim edilecek dosyalar	`Makefile, libft.h, ft_*.c`
Makefile	`NAME, all, clean, fclean, re`
Harici fonksiyonlar.	`Detaylar aşağıda açıklanmış`
Libft kullanılabilir mi?	`n/a`
Açıklama	`Kendi kitaplığınızı yazın: kürsüsünüz için faydalı bir araç olacak bir fonksiyon koleksiyonu.`

III.1 Teknik Özellikler

- Global değişken tanımlamak yasaktır.
- Daha karmaşık bir fonksiyon bölmek için yardımcı fonksiyonlara ihtiyacınız varsa, onları `statik` olarak tanımlayın. Bu şekilde kapsamları kullanılan dosya ile sınırlandırılacaktır.,
- Tüm dosyalarınızı deponuzun kök dizinine yerleştirin.
- Kullanılmayan dosya yüklemek yasaktır.
- Her `.c` dosyası, `-Wall -Wextra -Werror` bayrakları ile derlenmelidir.
- Kitaplığınızı oluşturmak için `ar` komutunu kullanmalısınız. `libtool` komutunun kullanılması yasaktır.
- `libft.a` dosyanız, havuzunuzun kökünde oluşturulmalıdır.

III.2 Bölüm 1 - Libc Fonksiyonları

İlk olarak, `libc`'den bir dizi fonksiyon yeniden yazmalısınız. Fonksiyonlar aynı prototiplere sahip olacak ve orijinallerle aynı davranışları uygulayacaktır. `man`'da tanımlanan şeklinde uymaları gerekir. Tek fark isimleri olacak. '`ft_`' önekiyle başlayacaklar. Örneğin, `strlen`, `ft_strlen` olacak.

Baştan yazmanız gereken bazı foksiyon prototipleri 'restrict' niteleyicisini kullanır. Bu anahtar kelime, c99. Bu nedenle, onu kendi prototiplerinize dahil etmeniz ve kodunuzu -std=c99 bayrağıyla derlemeniz yasaktır.

Aşağıdaki fonksiyonları 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 yazmak için `malloc()` kullanmanız gerekir:

- `calloc`
- `strdup`

III.3 Bölüm 2 - Ekstra Fonksiyonlar

İkinci bölümde, ya `libc`'de olmayan ya da parçalı ama farklı bir biçimde olan bir dizi fonksiyon geliştirmelisiniz.

Aşağıdaki fonksiyonlardan bazıları, 1. 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 döner.
Harici fonksiyonlar	`malloc`
Açıklama	(malloc(3) ile) hafıza ayırılır ve belirtilen substringi döner. Substring başlangıç '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:` Baş string. `s2:` Son string.
Return değeri	Yeni oluşturulan string. Eğer allocation hatası varsa NULL döner.
Harici fonksiyonlar	`malloc`
Açıklama	(malloc(3) ile) hafıza ayırılır ve çıktı olarak s1 ve s2 stringlerinin birleştirilmiş hali döndürülür.

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 döner.
Harici fonksiyonlar	malloc
Açıklama	(malloc(3) ile) hafıza ayırılır ve ardından başta ve sonda çıkartılmak istenilen 'set'teki karakterlersiz 's1' stringinin kopyası yaratılır.

Fonksiyon adı	ft_split
Prototip	char **ft_split(char const *s, char c);
Teslim edilecek dosyalar	-
Parametreler	s: Bölünecek string. c: Ayırıcı karakterler.
Return değeri	Bölünme sonucu elde edilen string dizisi. Eğer allocation hatası varsa NULL döner.
Harici fonksiyonlar	malloc, free
Açıklama	(malloc(3) ile) hafıza ayırılır ve ardından 's' stringini 'c' ayırıcı karakter ile bölerek yeni bir string dizisi dönülür. String dizisinin NULL pointer ile sonlanması gerekmektedir.

Fonksiyon adı	ft_itoa
Prototip	char *ft_itoa(int n);
Teslim edilecek dosyalar	-
Parametreler	n: Dönüştürülecek olan integer değeri.
Return değeri	String'e dönüşttürülen integer. Eğer allocation hatası varsa NULL döner.
Harici fonksiyonlar	malloc
Açıklama	(malloc(3) ile) hafıza ayırılır ve integer olarak alınan değerinini string'e döndürülür. Negatif sayılarda ele alınmalıdır.

Fonksiyon adı	ft_strmapi
Prototip	char *ft_strmapi(char const *s, char (*f)(unsigned int, char));
Teslim edilecek dosyalar	-
Parametreler	s: Üzerinde dolaşılacak string. f: Her bir karaktere uyugulanacak fonksiyon.
Return değeri	F fonksiyonun karakterlere uygulanması sonucu oluşturulan string. Eğer allocation hatası varsa NULL döner.
Harici fonksiyonlar	malloc
Açıklama	Karakterin indeksini ilk argüman olarak gönderip, 'f' fonksiyonunu 's' stringinin bütün karakterlerine uygular. Değiştirlen stringden yeni bir string yaratılır (malloc(3) ile).

Fonksiyon adı	ft_striteri
Prototip	void ft_striteri(char *s, void (*f)(unsigned int, char*));
Teslim edilecek dosyalar	-
Parametreler	s: Üzerinde dolaşılacak string. f: Her karatere uyugulanacak fonksiyon.
Return değeri	Yok
Harici fonksiyonlar	Yok
Açıklama	Karakterin indeksini ilk argüman olarak gönderip, 'f' fonksiyonunu 's' stringinin bütün karakterlerine uygular. Her karakterin adresi, karakteri değiştirmek için 'f' e gönderilmelidir

Fonksiyon adı	ft_putchar_fd
Prototip	void ft_putchar_fd(char c, int fd);
Teslim edilecek dosyalar	-
Parametreler	c: Yazılacak karakter. fd: Üzerine yazılacak olan file descriptor.
Return değeri	Yok
Harici fonksiyonlar	write
Açıklama	File descriptora 'c' karakterinin çıktısını yazar.

Fonksiyon adı	ft_putstr_fd
Prototip	void ft_putstr_fd(char *s, int fd);
Teslim edilecek dosyalar	-
Parametreler	s: Yazılacak string. fd: Üzerine yazılacak olan file descriptor.
Return değeri	Yok
Harici fonksiyonlar	write
Açıklama	File descriptora 's' stringinin çıktısını yazar.

Fonksiyon adı	ft_putendl_fd
Prototip	void ft_putendl_fd(char *s, int fd);
Teslim edilecek dosyalar	-
Parametreler	s: Yazılacak string. fd: Üzerine yazılacak olan file descriptor.
Return değeri	Yok
Harici fonksiyonlar	write
Açıklama	File descriptora 's' stringinin çıktısını yeni satır ekleyerek yazar.

Fonksiyon adı	ft_putnbr_fd
Prototip	void ft_putnbr_fd(int n, int fd);
Teslim edilecek dosyalar	-
Parametreler	n: Yazılacak integer. fd: Üzerine yazılacak olan file descriptor.
Return değeri	Yok
Harici fonksiyonlar	write
Açıklama	'n' integer değerinin çıktısını verilen file descriptora yazar.

Bölüm IV

Bonus Kısım

Zorunlu kısmı tamamladıysanız, bu ekstra kısmı yaparak daha ileri gitmekten çekinmeyin. Başarıyla geçilirse bonus puan getirecektir.

Bellek ve dizeleri işlemek için fonksiyonlar çok kullanışlıdır. Ancak daha yakın zamanda listeleri manipüle etmenin daha da yararlı olduğunu keşfedeceksiniz.

Listenizin bir elemanı temsil etmek için aşağıdaki yapıyı kullanmanız gerekir. 'Declaration'u `libft.h` dosyanıza ekleyin:

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

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

- `content` : Elementin içerdiği veri. `void *` tipi her tür veriyi tutmanızı sağlar.
- `next`: Bir sonraki elementin adresini tutar. Eğer son elemensa `NULL` değerindedir.

Makefile'da `make bonus` komutu bonus fonksiyonlarını libft.a kütüphanesine ekleyecektir.

Bonus kısmı, yalnızca zorunlu kısım MÜKEMMEL ise değerlendirilecektir. Mükemmel, zorunlu kısmın komple olarak yapıldığı ve arızasız çalıştığı anlamına gelir. TÜM zorunlu gereksinimleri geçmediyseniz, bonus bölümünüz hiç değerlendirilmeyecektir.

Listeleri kolaylıkla manipüle etmenizi sağlayacak aşağıdaki fonksiyonları ekleyin.

Fonksiyon adı	`ft_lstnew`
Prototip	`t_list *ft_lstnew(void *content);`
Teslim edilecek dosyalar	-
Parametreler	`content:` Yeni element oluşturacağınız content değişkeni.
Return değeri	Yeni element
Harici fonksiyonlar	`malloc`
Açıklama	(malloc(3) ile) hafıza ayırılır ve yeni element çıktı olarak verilir. Element'ın 'content' değişkeni parametredeki 'content' değeri ile başlatılır. Next değişkeni ise NULL değeri ile başlatılmalıdır.

Fonksiyon adı	`ft_lstadd_front`
Prototip	`void ft_lstadd_front(t_list **lst, t_list *new);`
Teslim edilecek dosyalar	-
Parametreler	`lst:` Listenin ilk elemanının pointer adresi. `new:` Listeye ekelenecek olan elemanın adresi.
Return değeri	Yok
Harici fonksiyonlar	Yok
Açıklama	Listenin başına yeni bir 'new' elemanı 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 eleman 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 elemanı
Harici fonksiyonlar	Yok
Açıklama	Listenin son elemanı döner.

Fonksiyon adı	ft_lstadd_back
Prototip	void ft_lstadd_back(t_list **lst, t_list *new);
Teslim edilecek dosyalar	-
Parametreler	lst: Listenin ilk elemanının pointer adresi. new: Listeye ekelenecek olan elemanın adresi.
Return değeri	Yok
Harici fonksiyonlar	Yok
Açıklama	Listenin sonuna yeni bir 'new' elemanı ekler.

Fonksiyon adı	ft_lstdelone
Prototip	void ft_lstdelone(t_list *lst, void (*del)(void *));
Teslim edilecek dosyalar	-
Parametreler	lst: Free edilecek eleman. del: İçeriği silmek için kullanılacak fonksiyonun adresi.
Return değeri	Yok
Harici fonksiyonlar	free
Açıklama	Parametre olarak bir eleman alır ve parametre olarak verilen 'del' fonksiyonunu kullanarak elemanın 'content' ini ve elemanı free. 'Next' inin hafızası free edilmemelidir.

Fonksiyon adı	ft_lstclear
Prototip	void ft_lstclear(t_list **lst, void (*del)(void *));
Teslim edilecek dosyalar	-
Parametreler	lst: Listedeki herangi bir elemanın pointerının adresi. del: İçeriği silmek için kullanılacak fonksiyonun adresi.
Return değeri	Yok
Harici fonksiyonlar	free
Açıklama	'del' ve free(3) kullanarak elemanı ve ona bağlı olan bütün elemanları siler ve hafızadaki yerini temizler. Son olarak listenin pointerı NULL' a ayarlanmalıdır.

Fonksiyon adı	ft_lstiter
Prototip	void ft_lstiter(t_list *lst, void (*f)(void *));
Teslim edilecek dosyalar	-
Parametreler	lst: Bir elemanın adresi. f: Listenin içerisinde gezinmek için kullanılacak olan fonksyionun adresi.
Return değeri	Yok
Harici fonksiyonlar	Yok
Açıklama	Listenin üzerinde dolanır ve 'f' fonksiyonunu listenin her elemanının içeriğine 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 elemanın adresi.`
	`f: Listenin içerisinde gezinmek için kullanılacak olan fonksiyonun adresi.`
	`del: Gerekli olduğunda elemanın 'content' ini temizlemeye yardımcı olan fonksiyonun adresi.`

Return değeri	`Yeni liste.` `Eğer allocation hatası olursa NULL döner.`
Harici fonksiyonlar	`malloc, free`
Açıklama	`'lst' listesi üzerinde dolaşır ve 'f' fonksiyonunu listenin her elemanına uygular. Uygulama sonucunda oluşan yeni elemenlardan yeni bir liste oluşturulur. Gerekli olduğu durumlarda 'del' fonksiyonu kullanıalarak elemanın 'content'i temizlenebilir.`

Bölüm V

Gönderme ve peer-evaluation

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

Tüm dosyalarınızı deponuzun kök dizinine yerleştirin.