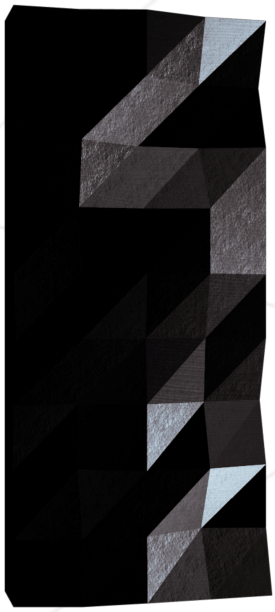

Libft

Kendi yazdığınız ilk kütüphane

Özet: Bu projenin amacı bir sonraki projelerinizde de kulalnabileceğiniz bir `C` kütüphanesi yazmaktır.

Versiyon: 15

İçindekiler

Bölüm I

Giriş

`C` programlama çok kullanışlı olan standart fonksiyonlara erişim olmadığı durumlarda çok sıkıcı olabilmektedir. Bu projede çok kullanışlı olan fonksiyonları tekrardan yazacak, çalışma mantıklarını anlayacak, bunları kullanmayı öğreneceksiniz. Baştan yazmış olduğunuz bu kütüphane gelecekteki `C` projeleriniz için çok faydalı olacaktır.

Bütün bir yıl boyunca `libft` projenizi geliştireceksiniz, eklemeler yapacaksınız. Bu süreçte hangi fonksiyonları kullanmaya izniniz olduğunu kontrol etmeyi 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	`*.c, libft.h, Makefile`
Makefile	`Evet`
Harici fonksiyonlar.	`Detaylar aşağıda açıklanmış`
Libft kullanılabilir mi?	`Kendiniz oluşturacaksınız`
Açıklama	`Cursusunuz için önemli fonksiyonları içeren kendi kütüphanenizi yazın`

III.1 Teknik Hususlar

- Global değişken tanımlamak yasaktır.
- Eğer karmaşık fonksiyonları yazmak için alt fonksiyonlara ihtiyacınız varsa bu alt fonksiyonları `static` olarak tanımlamanız gerekmektedir. Bunun nedeni bu fonksiyonları kütüphanenizle birlikte yayınlamamanızı sağlamaktadır. Bunu gelecekteki projeleriniz için de uygulamanız sizin yararınıza olacaktır.
- Reponuzun rootundaki bütün dosyaları yükleyin.
- Kullanılmayan dosyaların yüklenmesi yasaktır.
- Bütün .c dosyaları flagler ile derlenmelidir.
- Kendi kütüphanenizi oluşturmak için ar komutunu kullanmanız gerekmektedir. `Libtool` kullanılması yasaktır.

III.2 Bölüm 1 - Libc Fonksiyonları

Birinci bölümde, bazı `libc` fonksiyonlarını teknik dokümanlarına göre baştan yazacaksınız. Baştan yazmış olduğunuz fonksiyon orijinali ile aynı prototipi ve davranışı yansıtmalıdır. Fonksiyonlarınızın adı "`ft_`" ön ekini içermelidir. Örneğin, `strlen` fonksiyonunu baştan yazıldığında `ft_strlen` halini alacaktır.

Baştan yazacağınız bazı fonksiyonların prototipleri "restrict" etiketini kullanır. "restrict" etiketi c99 standartının bir parçasıdır. Bu yüzden "restrict" etiketini kendi projelerinize eklemek ve -std=c99 flagini kullanarak derlemek 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`

Aynı zamanda aşağıdaki fonksiyonları da `malloc` kullanarak tekrar yazmanız gerekmektedir.

- `calloc`
- `strdup`

III.3 Bölüm 2 - Ekstra Fonksiyonlar

İkinci bölümde `libc` kütüphanesinin içerisinde olmayan veya bu kütüphane içerisine farklı bir formda dahil edilmiş bazı fonksiyonları baştan yazacaksınız. Bu fonksiyonlardan bazıları birinci bölümdeki fonksiyonları yazmak için faydalı olacaktır.

Fonksiyon adı	`ft_substr`
Prototip	`char *ft_substr(char const *s, unsigned int start, size_t len);`
Teslim edilecek dosyalar	-
Parametreler	#1. Substringin oluşturalacağı string. #2. Substringin ana string içerisindeki başlangıç indisi. #3. Substringin maksimum uzunluğu.
Return değeri	Substring. Eğer allocation hatası varsa NULL dönsün.
Harici fonksiyonlar	malloc
Açıklama	Malloc fonksiyonu kullanılarak memoryde hafıza ayırılır ve belirtilen substringi döner. Substring başlangıç indisinden başlar ve maksimum boyutuna kadar devam eder.

Fonksiyon adı	`ft_strjoin`
Prototip	`char *ft_strjoin(char const *s1, char const *s2);`
Teslim edilecek dosyalar	-
Parametreler	#1. Ön ek stringi #2. Son ek stringi
Return değeri	Yeni oluşturulan string. Eğer hafıza ayırmada problem varsa NULL dönecek.
Harici fonksiyonlar	malloc
Açıklama	Malloc kullanrak hafızadan bir parça 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	#1. Kırpılacak string. #2. Kırpılması istenen karakterler.
Return değeri	Kırpılmış string. Eğer allocation hatasında problem varsa NULL dönecek.
Harici fonksiyonlar	malloc
Açıklama	Malloc kullanrak hafızada yer ayırılır ardından ana stringde kırpılmak istenilen karakterlerin hepsi kırpılır sonuç olarak elde edilen yeni string döndürülür.

Fonksiyon adı	ft_split
Prototip	char **ft_split(char const *s, char c);
Teslim edilecek dosyalar	-
Parametreler	#1. Bölünecek string. #2. Ayırıcı karakterler.
Return değeri	Ayırma sonucu elde edilen stringler. Eğer allocation hatası olursa NULL dönülecek.
Harici fonksiyonlar	malloc, free
Açıklama	Malloc kullanılarak hafızada yer ayırılır ardından verilmiş olan ayırıcı karakter yardımı ile string parçalara ayırılır ve bu yeni stringler dönülür. Stringlerin NULL pointer ile sonlanması gerekmektedir.

Fonksiyon adı	ft_itoa
Prototip	char *ft_itoa(int n);
Teslim edilecek dosyalar	-
Parametreler	#1. Dönüştürülecek olan integer değeri.
Return değeri	Integer değerini temsil eden string. Eğer allocation hatası olursa NULL dönecek.
Harici fonksiyonlar	malloc
Açıklama	Malloc kullanarak hafızada yer ayırılır ardından integer değerini temsil eden string döndürülür. Negatif sayılar negatif olarak döndürülmelidir.

Fonksiyon adı	ft_strmapi
Prototip	char *ft_strmapi(char const *s, char (*f)(unsigned int, char));
Teslim edilecek dosyalar	-
Parametreler	#1. Üzerinde dolaşılacak string değeri. #2. Her bir karaktere uyugulanacak fonksiyon.
Return değeri	F fonksiyonun karakterlere uygulanması sonucu oluşturulan string. Eğer allocation hatası olursa NULL değeri döndürülecek.
Harici fonksiyonlar	malloc
Açıklama	'f' fonksiyonunu 's' stringinin bütün karakterlerine uygular. Değiştirlen stringden yeni bir string yaratılır.

Fonksiyon adı	ft_striteri
Prototip	void ft_striteri(char *s, void (*f)(unsigned int, char*));
Teslim edilecek dosyalar	-
Parametreler	#1. Üzerinde dolaşılacak string değeri. #2. Her karatere uyugulanacak fonksiyon.
Return değeri	None.
Harici fonksiyonlar	None
Açıklama	'f' fonksiyonun stringin her karakterine uygular. Eğer gerekli olursa her karakter adresi ile gönderilmelidir

Fonksiyon adı	ft_putchar_fd
Prototip	void ft_putchar_fd(char c, int fd);
Teslim edilecek dosyalar	-
Parametreler	#1. Çıkış karakteri. #2. Üzerine yazılacak olan file descriptor.
Return değeri	None
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	#1. Çıkışa verilecek string. #2. Yazılacak olan file descriptor.
Return değeri	None
Harici fonksiyonlar	write
Açıklama	's' stringini verilen file descriptor içerisine yazar.

Fonksiyon adı	ft_putendl_fd
Prototip	void ft_putendl_fd(char *s, int fd);
Teslim edilecek dosyalar	-
Parametreler	#1. Çıkışa verilecek string. #2. Yazılacak olan file descriptor.
Return değeri	None
Harici fonksiyonlar	write
Açıklama	's' string çıktısını sonunda new line karakteri ile birlikte verilen file descriptora yazar.

Fonksiyon adı	ft_putnbr_fd
Prototip	void ft_putnbr_fd(int n, int fd);
Teslim edilecek dosyalar	-
Parametreler	#1. Çıkışa verilecek integer değeri. #2. Yazılacak olan file descriptor.
Return değeri	None
Harici fonksiyonlar	write
Açıklama	Integer 'n' değerinin çıktısını verilen file descriptora yazar.

Bölüm IV

Bonus Kısım

Eğer zorunlu kısmı tamamlamış iseniz, geri kalan kısımda eğlenmenize bakabilirsiniz. Anlayabileceğiniz üzere son kısım bonus puanları alabileceğiniz kısımdır.

Memoryi ve stringleri manipüle edebilen fonksiyonlara sahip olmak çok kullanışlıdır fakat listeleri manipüle edebilen fonksiyonlara sahip olmanın çok daha kullanışlı olduğunu yakında göreceksiniz.

`make bonus` komutu bonus fonksiyonlarını libft.a kütüphanesine ekelyecektir.

Listenizin elementlerini temsil etmek için takip eden yapıyı kullanacaksınız. Bu yapı mutlaka sizin `libft.h` dosyanızın içerisine eklenmelidir.

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

Aşağıda `t_list` structının bileşenlerinin tanımı yapılmaktadır. struct:

- `content` : Elementin içerdiği veri. `void *` tipinde olması istediğiniz türde veriyi tutmanızı sağlar.
- `next` : Bir sonraki elementin adresini tutar eğer son elemensa `NULL` değerindedir.

Aşağıdaki fonksiyonlar listelerinizi kolaylıkla manipüle etmenizi sağlayacaktır.

Fonksiyon adı	`ft_lstnew`
Prototip	`t_list *ft_lstnew(void *content);`
Teslim edilecek dosyalar	-
Parametreler	#1. Yeni element oluşturacağınız content değişkeni.
Return değeri	Yeni element.
Harici fonksiyonlar	malloc
Açıklama	Malloc kullanarak memoryden yer ayırılır ve yeni element çıktı olarak verilir. Content değişkeni 'content' parametresinin 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	#1. Listenin ilk bağlantısının pointerının adresi. #2. Listeye ekelenecek olan elemanın pointerının adresi.
Return değeri	None
Harici fonksiyonlar	None
Açıklama	Listenin başına yeni bir eleman ('new') ekler.

Fonksiyon adı	`ft_lstsize`
Prototip	`int ft_lstsize(t_list *lst);`
Teslim edilecek dosyalar	-
Parametreler	#1. Listenin başlangıcı.
Return değeri	Listenin boyunu dönmelidir.
Harici fonksiyonlar	None
Açıklama	Listedeki eleman sayısını bulur.

Fonksiyon adı	`ft_lstlast`
Prototip	`t_list *ft_lstlast(t_list *lst);`
Teslim edilecek dosyalar	-
Parametreler	#1. Listenin başlangıcı.
Return değeri	Listenin son elemanı.
Harici fonksiyonlar	None
Açıklama	Listenin son elemanın döner.

Fonksiyon adı	`ft_lstadd_back`
Prototip	`void ft_lstadd_back(t_list **lst, t_list *new);`
Teslim edilecek dosyalar	-
Parametreler	#1. Listenin ilk elemanın pointerının adresini döner. #2. Listeye eklenecek olan elemanın adresini döner.
Return değeri	None
Harici fonksiyonlar	None
Açıklama	'new' elemanını listenin en sonuna ekler.

Fonksiyon adı	`ft_lstdelone`
Prototip	`void ft_lstdelone(t_list *lst, void (*del)(void *));`
Teslim edilecek dosyalar	-
Parametreler	#1. Free edilecek eleman. #2. İçeriği silmek için kullanılacak fonksiyonun adresi.
Return değeri	None
Harici fonksiyonlar	free
Açıklama	Parametre olarak bir eleman alır ve fonksyiona bir diğer paramtere olarak verilmiş olan 'del' fonksiyonunu kullanarak elemanın memorydeki yerini temizler. 'Next' in memorydeki yeri temizlenmiş olmamalıdır.

Fonksiyon adı	ft_lstclear
Prototip	void ft_lstclear(t_list **lst, void (*del)(void *));
Teslim edilecek dosyalar	-
Parametreler	#1. Listedeki herangi bir elemanın pointerının adresi. #2. İçeriği silmek için kullanılacak fonksiyonun adresi.
Return değeri	None
Harici fonksiyonlar	free
Açıklama	'del' ve free(3) kullanarak elemanı siler ve hafızadaki yerini temizler. Ayrıca silme işleminde elmanın tüm successorlarını da temizler. Sonuç olarak listenin pointerı NULL' a ayarlanmalıdır.

Fonksiyon adı	ft_lstiter
Prototip	void ft_lstiter(t_list *lst, void (*f)(void *));
Teslim edilecek dosyalar	-
Parametreler	#1. Listedeki herangi bir elemanın pointerının adresi. #2. Listenin içerisinde gezinmek için kullanılacak olan fonksyionun adresi.
Return değeri	None
Harici fonksiyonlar	None
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	#1. Listedeki herangi bir elemanın pointerının adresi. #2. Listenin içerisinde gezinmek için kullanılacak olan fonksyionun 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 eleamanına uygular. Uygulama sonucunda oluşan yeni elemenlardan yeni bir liste oluşturulur. Gerekli olduğu durumlarda delete fonksiyonu kullanıalarak elemanın içeriği temizlenebilir. #2. Listenin içerisinde gezinmek için kullanılacak olan fonksiyonun adresi. #3. Gerekli olduğunda elemanın içeriğini temizlemeye yardımcı olan fonksiyonun adresi.