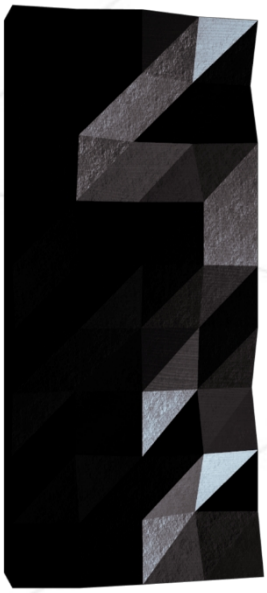

Sonraki Satırı Al

Bir fd'den bir satır okumak çok sıkıcıdır

Özet:
Bu proje, bir dosya tanımlayıcısından okunan bir satırı döndüren bir işlevi programlamakla ilgilidir.

Sürüm: 11

İçindekiler

Bölüm I

Hedefler

Bu proje sadece koleksiyonunuza çok kullanışlı bir fonksiyon eklemenize izin vermekle kalmayacak, aynı zamanda C programlamada oldukça ilginç yeni bir kavramı öğrenmenizi sağlayacaktır: statik değişkenler.

Bölüm II

Ortak Talimatlar

- Projeniz C dilinde yazılmış olmalıdır.
- Projeniz Norm'a uygun olarak yazılmış olmalıdır. Eğer bonus dosyalarınız/fonksiyonlarınız varsa bunlar norm kontrolüne dahil edilir ve içinde norm hatası varsa 0 alırsınız.
- Fonksiyonlarınız, tanımlanmamış davranışlar dışında beklenmedik bir şekilde (segmentasyon hatası, veri yolu hatası, çift serbest bırakma vb.) çıkmamalıdır. Böyle bir durumda, projeniz işlevsel değil olarak kabul edilecek ve değerlendirme sırasında 0 alacaktır.
- Tüm heap tahsisli bellek alanı gerektiğinde uygun şekilde serbest bırakılmalıdır. Hiçbir sızıntıya müsamaha gösterilmeyecektir.
- Konu gerektiriyorsa, kaynak dosyalarınızı -Wall, -Wextra ve -Werror bayraklarıyla gerekli çıktıya derleyecek bir Makefile sunmanız, cc kullanmanız ve Makefile'ınızın yeniden bağlanmaması gerekir.
- Makefile dosyanız en azından $(NAME), all, clean, fclean kurallarını içermeli ve re.
- Projenize bonuslar eklemek için, Makefile dosyanıza bir kural bonusu eklemelisiniz; bu bonus, projenin ana bölümünde yasak olan tüm çeşitli başlıkları, kütüphaneleri veya işlevleri ekleyecektir. Konu başka bir şey belirtmiyorsa bonuslar farklı bir _bonus.{c/h} dosyasında olmalıdır. Zorunlu ve bonus kısım değerlendirmesi ayrı ayrı yapılır.
- Projeniz libft kullanmanıza izin veriyorsa, kaynaklarını ve ilişkili Makefile'ını ilişkili Makefile'ı ile birlikte bir libft klasörüne kopyalamanız gerekir. Projenizin Makefile'ı kütüphaneyi Makefile'ını kullanarak derlemeli, ardından projeyi derlemelidir.
- Bu çalışma **teslim edilmek zorunda olmasa ve notlandırılmasa** bile projeniz için test programları oluşturmanızı teşvik ediyoruz. Bu size kendi çalışmanızı ve meslektaşlarınızın çalışmalarını kolayca test etme şansı verecektir. Bu testleri özellikle savunmanız sırasında faydalı bulacaksınız. Nitekim, savunma sırasında kendi testlerinizi ve/veya değerlendirdiğiniz akranınızın testlerini kullanmakta özgürsünüz.
- Çalışmanızı size atanmış git deposuna gönderin. Yalnızca git deposundaki çalışmalara not verilecektir. Çalışmanıza not vermek için Deepthought görevlendirilirse, bu işlem şu şekilde yapılacaktır

akran değerlendirmelerinizden sonra. Deepthought'un notlandırması sırasında çalışmanızın herhangi bir bölümünde bir hata meydana gelirse, değerlendirme duracaktır.

Bölüm III Zorunlu kısım

İşlev adı	get_next_line
Prototip	char *get_next_line(int fd);
Dosyaları teslim edin	get_next_line.c, get_next_line_utils.c, get_next_line.h
Parametreler	fd: Okunacak dosya tanımlayıcısı
Dönüş değeri	Satırı oku: doğru davranış NULL: okunacak başka bir şey yok veya bir hata oluştu
Harici fonksiyonlar.	read, malloc, free
Açıklama	'den okunan bir satırı döndüren bir fonksiyon yazın. dosya tanımlayıcısı

- get_next_line() işlevinize tekrarlanan çağrılar (örneğin, bir döngü kullanarak), dosya tanımlayıcısının işaret ettiği metin dosyasını **her seferinde bir satır** okumanıza izin vermelidir.
- İşleviniz okunan satırı döndürmelidir.
  Okunacak başka bir şey yoksa veya bir hata oluştuysa, NULL döndürmelidir.
- İşlevinizin hem bir dosyayı okurken hem de standart girdiden okurken beklendiği gibi çalıştığından emin olun.
- Dosya sonuna ulaşıldığı ve \n karakteri ile bitmediği durumlar hariç, döndürülen satırın sonlandırıcı *\n* karakterini içermesi gerektiğini **lütfen unutmayın.**
- get_next_line.h başlık dosyanız en azından şu prototipi içermelidir get_next_line() fonksiyonu.
- İhtiyacınız olan tüm yardımcı fonksiyonları get_next_line_utils.c dosyasına ekleyin.

Statik değişkenin ne olduğunu bilmek iyi bir başlangıç olacaktır.

- get_next_line() işlevinde dosyaları okumanız gerekeceğinden, derleyici çağrınıza bu seçeneği ekleyin: -D BUFFER_SIZE=n
  read() için tampon boyutunu tanımlayacaktır.
  Tampon boyutu değeri, kodunuzu test etmek için eş-değerlendiricileriniz ve Moulinette tarafından değiştirilecektir.

Bu projeyi normal bayraklara ek olarak -D BUFFER_SIZE bayrağı ile ve bu bayrak olmadan derleyebilmeliyiz. İstediğiniz varsayılan değeri seçebilirsiniz.

- Kodunuzu aşağıdaki gibi derleyeceksiniz (örnek olarak 42'lik bir tampon boyutu kullanılmıştır):
  cc **-Wall** -Wextra -Werror -D BUFFER_SIZE=42 <files>.c
- Dosya tanımlayıcısının işaret **ettiği** dosya son çağrıdan bu yana değiştiyse get_next_line() işlevinin tanımlanmamış bir davranışa sahip olduğunu, read() işlevinin ise dosyanın sonuna ulaşmadığını düşünüyoruz.
- Ayrıca get_next_line() işlevinin ikili bir dosyayı okurken tanımlanmamış bir davranışa sahip olduğunu düşünüyoruz. Ancak, isterseniz bu davranışı işlemek için mantıklı bir yol uygulayabilirsiniz.

BUFFER_SIZE değeri 9999 ise fonksiyonunuz hala çalışıyor mu? Eğer 1 ise? 10000000? Nedenini biliyor musunuz?

get_next_line() her çağrıldığında mümkün olduğunca az okumaya çalışın.
Yeni bir satırla karşılaşırsanız, geçerli satırı döndürmeniz gerekir.
Tüm dosyayı okumayın ve ardından her satırı işlemeyin.

Yasak

- Bu projede **libft**'inizi kullanmanıza izin verilmez.
- lseek() yasaklanmıştır.
- Global değişkenler yasaktır.

Bölüm IV

Bonus kısmı

Bu proje basittir ve karmaşık bonuslara izin vermez. Ancak, yaratıcılığınıza güveniyoruz. Zorunlu bölümü tamamladıysanız, bu bonus bölümünü deneyin.

İşte bonus bölüm gereksinimleri:

- get_next_line() işlevini yalnızca bir statik değişken kullanarak geliştirin.
- get_next_line() işleviniz aynı anda birden fazla dosya tanımlayıcısını yönetebilir. Örneğin, 3, 4 ve 5 numaralı dosya tanımlayıcılarından okuyabiliyorsanız, her dosya tanımlayıcısının okuma iş parçacığını kaybetmeden veya başka bir fd'den bir satır döndürmeden çağrı başına farklı bir fd'den okuyabilmeniz gerekir.
  Bu, fd 3'ten okumak için get_next_line() işlevini çağırabilmeniz gerektiği anlamına gelir, sonra fd 4, sonra 5, sonra bir kez daha 3, bir kez daha 4 ve bu şekilde devam eder.

Bonus parça dosyalarına _bonus.[*c**h*] son ekini ekleyin.
Bu, zorunlu parça dosyalarına ek olarak aşağıdaki 3 dosyayı da teslim edeceğiniz anlamına gelmektedir:

- get_next_line_bonus.c
- get_next_line_bonus.h
- get_next_line_utils_bonus.c

Bonus kısım sadece zorunlu kısım MÜKEMMEL ise değerlendirilecektir. Mükemmel, zorunlu kısmın bütünsel olarak yapıldığı ve hatasız çalıştığı anlamına gelir. TÜM zorunlu gereklilikleri geçmediyseniz, bonus bölümünüz hiç değerlendirilmeyecektir.

Bölüm V

Sunum ve akran değerlendirmesi

Ödevinizi her zamanki gibi Git deponuzda teslim edin. Savunma sırasında yalnızca deponuzdaki çalışmalar değerlendirilecektir. Doğru olduklarından emin olmak için dosyalarınızın adlarını iki kez kontrol etmekten çekinmeyin.

Testlerinizi yazarken şunu unutmayın:
1) Hem tampon boyutu hem de satır boyutu çok farklı değerlerde olabilir.
2) Bir dosya tanımlayıcısı yalnızca normal dosyalara işaret etmez.
Akıllı olun ve meslektaşlarınızla çapraz kontrol yapın. Savunma için çeşitli testlerden oluşan eksiksiz bir set hazırlayın.

Geçtikten sonra, get_next_line() işlevini libft'inize eklemekten çekinmeyin.