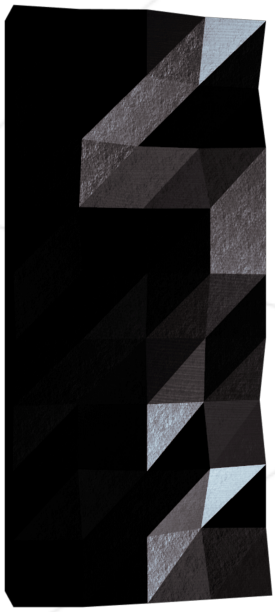

Pipex

Özet:
Bu proje zaten bildiğiniz bir UNIX mekanizmasını programınızda kullandırtarak daha detaylı keşfetmenizi sağlayacak.

Versiyon: 3

İçindekiler

Bölüm I

Ön Söz

Cristina: "Gidin bir yerlerde salsa dansı yapın :)"

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	`pipex`
Teslim edilecek dosyalar	`Makefile, *.h, *.c`
Makefile	`NAME, all, clean, fclean, re`
Argümanlar	`file1 cmd1 cmd2 file2`
Harici fonksiyonlar.	• `open, close, read, write, malloc, free, perror, strerror, access, dup, dup2, execve, exit, fork, pipe, unlink, wait, waitpid` • `ft_printf veya SİZİN kodladığınız bir alternatif`
Libft kullanılabilir mi?	`Evet`
Açıklama	`Bu proje pipeları yönetmek ile ilgilidir.`

Programınız aşağıdaki gibi çalıştırılacaktır:

```
./pipex file1 cmd1 cmd2 file2
```

Programınız 4 argüman almalıdır:

- `file1` ve `file2` **dosya isimleridir**.
- `cmd1` ve `cmd2` parametreleri ile **shell komutlarıdır.**

Aşağıdaki shell komutu ile birebir aynı davranmalıdır:

```
$> < file1 cmd1 | cmd2 > file2
```

III.1 Örnekler

```
$> ./pipex infile "ls -l" "wc -l" outfile
```

Bu şekilde çalışmalıdır: `< infile ls -l | wc -l > outfile`

```
$> ./pipex infile "grep a1" "wc -w" outfile
```

Bu şekilde çalışmalıdır: `< infile grep a1 | wc -w > outfile`

III.2 Gereksinimler

Projeniz aşağıdaki kurallara uymalıdır.

- Source dosyalarınızı compile edecek bir `Makefile` dosyası oluşturmalı ve proje repositorynize yüklemelisiniz. Makefile relink yapmamalıdır.
- Oluşabilecek hataları detaylıca handle etmelisiniz. Programınız beklenmedik bir şekilde çıkış yapmamalıdır (segmentasyon hatası, bus error, çifte free, ve benzeri.)
- Programınızda **memory leakleri** bulunmamalıdır.
- Eğer herhangi bir şüpheniz varsa, hata yönetimini aşağıdaki shell komutu ile test edin:
  `< file1 cmd1 | cmd2 > file2`

Bölüm IV

Bonus Bölüm

Eğer aşağıdakileri yaparsanız ekstra puan kazanacaksınız:

- Birden fazla pipe'ı handle edin.

  Bu komut:

  ```
  $> ./pipex file1 cmd1 cmd2 cmd3 ... cmdn file2
  ```

  Bu komut gibi çalışmalıdır:

  ```
  < file1 cmd1 | cmd2 | cmd3 ... | cmdn > file2
  ```

- "here_doc" ilk parametresiyken « ve » redirectionlarını handle edin.

  Bu komut:

  ```
  $> ./pipex here_doc LIMITER cmd cmd1 file
  ```

  Bu komut gibi çalışmalıdır:

  ```
  cmd << LIMITER | cmd1 >> file
  ```

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.

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.

file.bfe:VACsSfsWN1cy33ROeASsmsgnY0o0sDMJev7zFHhw
QS8mvM8V5xQQpLc6cDCFXDWTiFzZ2H9skYkiJ/DpQtnM/uZ0